Language Modeling for Limited-Data Domains

by

Bo-June (Paul) Hsu

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2009

Author .....
Department of Electrical Engineering and Computer Science
February 27, 2009

Certified by .....
James R. Glass
Principle Research Scientist
Thesis Supervisor

Accepted by .....
Terry P. Orlando
Chairman, Department Committee on Graduate Students

Language Modeling for Limited-Data Domains

by

Bo-June (Paul) Hsu

Submitted to the Department of Electrical Engineering and Computer Science
on February 27, 2009, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

With the increasing focus of speech recognition and natural language processing applications on domains with limited amount of in-domain training data, enhanced system performance often relies on approaches involving model adaptation and combination. In such domains, language models are often constructed by interpolating component models trained from partially matched corpora. Instead of simple linear interpolation, we introduce a generalized linear interpolation technique that computes context-dependent mixture weights from features that correlate with the component confidence and relevance for each n-gram context. Since the n-grams from partially matched corpora may not be of equal relevance to the target domain, we propose an n-gram weighting scheme to adjust the component n-gram probabilities based on features derived from readily available corpus segmentation and metadata to de-emphasize out-of-domain n-grams. In scenarios without any matched data for a development set, we examine unsupervised and active learning techniques for tuning the interpolation and weighting parameters.

Results on a lecture transcription task using the proposed generalized linear interpolation and n-gram weighting techniques yield up to a 1.4% absolute word error rate reduction over a linearly interpolated baseline language model. As more sophisticated models are only as useful as they are practical, we developed the MIT Language Modeling (MITLM) toolkit, designed for efficient iterative parameter optimization, and released it to the research community. With a compact vector-based n-gram data structure and optimized algorithm implementations, the toolkit not only improves the running time of common tasks by up to 40x, but also enables the efficient parameter tuning for language modeling techniques that were previously deemed impractical.

Thesis Supervisor: James R. Glass
Title: Principle Research Scientist

Acknowledgments

I would like to thank my advisor, Jim Glass, for the opportunities and support that I have received over the past four and half years at the Spoken Language Systems group. Thanks also go to the other members of my thesis committee, Regina Barzilay and Michael Collins, for their feedback on the research that led to this work. I would especially like to thank Regina for introducing me to the field of natural language processing and helping me learn the fundamentals of academic research when I first started at MIT.

The research for this thesis would not have been possible without the support from the entire SLS staff. Specifically, I would like to thank Lee Hetherington for assistance with SUMMIT and feedback on the MITLM toolkit, Chao Wang for constructive comments on early drafts of my papers, Scott Cyphers for help with the computing infrastructure, T. J. Hazen for guidance on the lecture recognizer, and Marcia Davidson for advice on logistics. Special thanks also go to Hung-an Chang for providing early feedback and assistance on various aspects of the thesis.

In addition to my thesis research, the past four and half years have been made more fulfilling with various sponsor projects, internship opportunities, and teaching experience. I am especially grateful to Victor Zue and Jim for providing me with the opportunity to work on the T-party Project and attend the meetings with Quanta Computer in Taiwan. I would also like to thank Milind Mahajan and Alex Acero in Microsoft, Brian Strope and Francoise Beaufays at Google, and Han Shu and Mike Phillips from vlingo for their mentorship during my summer internships there. Further acknowledgments go to the 2007 class of 6.345: *Automatic Speech Recognition* and the UROP and MEng students I have had the pleasure of working with over the past few years: Greg Yu, James Sun, David Koh, Jessie Li, Jay Patel, and Stanley Wang. I hope you have learned as much from the experience as I have.

Finally, I would like to thank my friends and family for their continual support and encouragement. In particular, I would like to thank my former and current office mates: Philip Bramsen, Chih-yu Chao, Chia-ying Lee, Igor Malioutov, Mitch Peabody, Pawan Deshpande, Yuan Shen, Gabi Zaccak, and Yaodong Zhang. The time I have spent at MIT has also been greatly enriched with my interactions with Stephen Hou, Jen Roberts, and other members of the Graduate Student Association. I would also like to thank Ingrid Bau for her unwaivering support over the past few years. Finally, I would like to thank my family for their encouragement to continually push myself and pursue my dreams.

This thesis is an expansion of previously published work [70, 72–75], based on research supported in part by the T-party Project, a joint research program between MIT and Quanta Computer Inc.; National Science Foundation under grant #IIS-0415865; and Nokia Corporation.

Paul
February, 2009

Contents

List of Figures

List of Tables

Vision

One of the main challenges in the development of spoken dialogue applications is the need for a large corpus of in-domain data in order to train, develop, and evaluate the various modules of the system, including the acoustic model, language model, natural language understanding component, and dialog manager. However, collecting data that accurately reflects realistic usage is difficult without a system that users can interact with [57].

To address this *chicken and egg* problem in data collection, some previous research has gathered data for the target application domain using a *Wizard of Oz* approach for initial system development [28, 53]. In this technique, users interact with a mock system controlled by human experimenters in the background. The behind-the-scenes experimenters not only transcribe the input spoken utterances, but also provide the most appropriate system responses. Although the data collected from such incognizant users reflects actual usage conditions, using *Wizard of Oz* techniques to collect the large amounts of in-domain data needed for model development is generally impractical.

As a result of the challenge in collecting large quantities of in-domain data, most speech application development tend to focus on popular domains, such as weather, travel, and restaurants, where in-domain data already exists or can be readily collected [57]. Specialized domains with smaller user bases, such as the MIT campus shuttle tracking information, are rarely considered even though such a system may be invaluable to the hundreds of students who ride it each day. In effect, we face another *chicken and egg* problem in speech application development. Users are unwilling to consider spoken user interfaces without sufficient domain coverage. However, for developers, it is not worth spending the effort on domains without sufficient users. Thus, the lack of diverse and personalized applications is one of the main hurdles to the adoption and popularity of spoken user interfaces.

Building effective models to accommodate a diverse group of users often requires significant amount of data to capture the complex variations observed across users. However, individual users often exhibit consistencies that can be captured with significantly simpler models. For example, in the data collected from the Jupiter weather information system [162], we have found more than 100 distinct phrases for requesting tomorrow's weather forecast for Boston. Yet, when asking individual subjects to generate distinct queries for the same task, most begin to hesitate after two or three sentences. In fact, it is precisely this consistency that makes it difficult for a single speech developer to author a grammar that covers all individual variations without performing large-scale data collection.

While individual consistency implies that the data collection effort should be distributed across a large number of participants to capture the diverse individual variations, it also suggests that to build a model for an individual user on a given task, it may be sufficient to train the model from only a few labeled examples from that individual. In many applications (see Figure 1), such labeled examples can actually be provided by the user as corrections or feedback to the system when the system makes a mistake with minimal additional effort.

Chinese Character Correction [71] – Correcting conversion errors with current phonetic input method editors is often painstaking and time consuming. By specifying the desired character via speech using character descriptions (言午許), we significantly reduce the effort needed to correct conversion errors. In situations where the spoken character descriptions are misrecognized or unknown, users can still rely on existing keyboard/mouse interfaces for correction, while providing valuable feedback for adaptation and personalization at the same time.

Multimodal Form Filling [76] – Form filling on many mobile devices is tedious and time-consuming due to the lack of a full keyboard and a small screen size. Using speech input, we can pre-populate the form fields with the speech recognition n-best hypotheses and simplify text entry to selecting the target item from a ranked list. If the desired item is not hypothesized, the user can fall back to the original text input method (soft keyboard, triple tap, T9, ...). By adapting the recognizer models with the committed field values, the system can learn user-specific carrier phrases and rapidly improve subsequent performance.

Text-to-Speech Pronunciation Correction – Text-to-speech systems require the correct pronunciation for each word synthesized. Since automatic pronunciation algorithms cannot predict the pronunciation for unknown words with perfect accuracy, the synthesized speech may contain mis-pronunciations. Instead of tolerating synthesis errors, users can verbally repeat the offending word with the correct pronunciation, similar to how children are taught to read. By recognizing these spoken pronunciation corrections, the system can learn the appropriate pronunciation of words in context.

Voice-Driven Web Navigation – With increasing amount of bookmarked web pages, finding a particular page has become so arduous that recent browsers have introduced search functionalities for bookmarks. Allowing users to access these bookmarked pages via voice commands requires correctly mapping the recognized command to a bookmark. Since users are motivated to complete the task, manually if necessary, we can utilize the subsequent user actions to gather training data and improve the mapping from speech to bookmarks.

Figure 1: Example applications with user feedback.

Thus, for simple tasks like correcting text-to-speech pronunciations, one example of a correct pronunciation is often sufficient.

By designing applications to rapidly adapt to an individual user via user feedback instead of using a fixed model that targets all users, we significantly reduce the amount of data needed to train a user-specific model. However, applications are generally composed of multiple tasks. Asking users to provide correction feedback for each of these tasks is likely to drive them away from spoken user interfaces, even if the system rapidly adapts. Fortunately, by pooling the data collected from a community of users for both model training and personalization, we effectively distribute the work needed to teach the system across all users.

In a collaborative environment like Wikipedia, over half of the edits are made by less than 0.7% of the registered users [147]. Similarly, we also expect a small fraction of enthusiasts of such a feedback-driven spoken dialog system to contribute the majority of the training data. Thus, assuming a healthy user base, the majority of the users will successfully complete common tasks without needing to provide any correction feedback. As users are more likely to tolerate mistakes in niche tasks, they are more willing to provide feedback to improve the system performance for the next time they access these tasks. Although individuals are primarily motivated through system personalization, the collective feedback indirectly yields better base models for all users and tasks. (See Appendix A for a more in-depth

discussion on collaborative data collection.)

By viewing spoken dialogue systems as learning novices instead of domain experts out of the box, we enable a community of users to teach and train the system through simple feedback. If the collection of user feedback is integrated with a natural user experience and leads to immediate improvements, users may be motivated to use spoken user interfaces and provide correction feedback when necessary. Through collaborative adaptation, we hope to increase application diversity and solve the *chicken and egg* challenge for application development.

In this thesis, we work towards the above vision by improving language modeling, as subsequent modules require an accurate transcription of the spoken utterance to effectively understand and respond to the user input. Since achieving application diversity implies working with a limited amount of in-domain data, we will specifically focus on language modeling for limited-data domains.

Chapter 1

Introduction

1.1 Language Modeling for Limited-Data Domains

Statistical models trained using machine learning techniques require a large amount of data from the target domain in order to learn the model parameters reliably. In fact, across a variety of natural language processing tasks, model performance continues to improve with increasing amounts of training data, even beyond one billion words [4, 11]. Thus, much recent research in natural language processing has been exploring techniques to utilize the World Wide Web and the Google 5-gram corpus [10] tabulated from one trillion words of web data to improve the model performance [15, 87, 145].

Building an effective language model also requires a large in-domain text corpus in order to accurately estimate the underlying word distributions. However, in many practical applications, such quantity of matched training data is not readily available. Despite the colossal size of the data available on the web, it is not an exact match for many application domains in terms of style and/or topic. In practice, many potentially valuable applications of language modeling have only limited amounts of matched training data. In extreme cases, no such data exists beyond the unlabeled test data itself.

Despite the scarcity of matched training data, text corpora that partially match the target domain in topic or style often exist in abundance. For instance, in query modeling for voice search [43, 99], we may only have a limited amount of transcribed voice search queries for building the speech recognition language model. However, we may have access to a large quantity of web search query logs that match the topics that users are likely to search for, although in a mismatched keyword search style. Likewise, it is relatively straightforward to collect closed caption transcripts from TV quiz shows that better match the voice search style, but not necessarily with the right topic distribution.

Take academic lecture transcription as another example. Lecture speech often exhibits a high degree of spontaneity and focuses on narrow topics with special terminologies [56]. For training, we may have existing transcripts from other lectures that match the spontaneous lecture style, but rarely on the target topic. We may also have text material on the topic in forms of the course textbook and lecture slides, but written language is a poor predictor of how words are actually spoken. However, unlike voice search, data that matches both the topic and style of the target lecture rarely exists.

In this thesis, instead of focusing on techniques that leverage web-scale data, we explore how to best utilize the often abundant partially matched corpora to improve language modeling performance in application domains with limited or no matched data.

1.2 Contributions

In domains with limited amounts of matched training data, the challenge in language modeling is how to best utilize partially matched corpora to model the target domain. For academic lectures, such data might include off-topic transcripts of general lectures and style-mismatched text from the course textbook. In this thesis, we will investigate multiple techniques to address the challenges in using partially matched data to build an effective n-gram language model for lecture speech transcription.

Generalized Linear Interpolation Traditional language model adaptation techniques generally interpolate n-gram models trained from these partially matched corpora using weights that are independent of the relevance of each component under different n-gram contexts. Intuitively, an on-topic course textbook component should better predict the word following the n-gram history OF THE than an off-topic general lectures component, as we are likely to observe a topic word within such context. Thus, we will present a generalized form of linear interpolation that models the interpolation weights as a log-linear function of n-gram history features indicative of the component relevance. As we will show, the resulting generalized linear interpolation approach subsumes both the traditional linear interpolation (LI) [81] and count merging (CM) [1] techniques as special cases. Using a combination of count and automatically derived topic features, generalized linear interpolation achieves up to a 0.8% and 0.3% absolute word error rate (WER) reduction over LI and CM, respectively.

n-gram Weighting Since the training corpora generally exhibit some degree of mismatch to the target domain in either topic or style, building component n-gram models directly from the text tends to assign disproportionately high probabilities to out-of-domain n-grams. To reduce such bias, we will propose an n-gram weighting technique that weights the count of each n-gram in a standard language model estimation procedure by a relevance factor computed from a log-linear combination of n-gram features. Using the readily available segmentation and metadata information associated with each corpus, we can derive features that correlate with the topic specificity of each n-gram. By combining n-gram weighting with model interpolation, we effectively de-emphasize the weights of out-of-domain n-grams and redistribute them to more relevant n-grams from other components. Compared with the LI and CM baselines, n-gram weighting reduces the absolute WER by up to 1.1% and 0.7%, respectively.

Parameter Optimization The above adaptation techniques require an in-domain development set for parameter estimation. However, in some application domains, no such data exist beyond the untranscribed test corpus. Therefore, we will explore the iterative use of the recognition hypotheses for unsupervised parameter estimation. In addition, we will evaluate the effectiveness of supervised adaptation using varying amounts of user-provided transcripts of utterances selected via multiple strategies. Although unsupervised parameter tuning achieves about 80% of the 1.1% supervised WER improvement, manually transcribing the least confident 300 words not only outperforms unsupervised adaptation, but also reduces the effective transcription WER by another 0.6%.

Data Structure & Algorithms Language modeling techniques are ultimately most useful if they can be efficiently implemented and applied without exorbitant computational

resources and significant human intervention. For example, although count merging generally outperforms linear interpolation, the lack of automatic tuning tools limits its usage, especially when combining three or more models. The same applies to modified Kneser-Ney smoothing with discount parameter tuning [22]. To ensure that the algorithms we develop can be practically applied, we will introduce a vector-based n-gram data structure designed for iterative parameter optimization and present efficient algorithms of not only the techniques presented here, but also ones previously considered difficult to optimize. Compared with the popular SRILM toolkit [138], we improved the running time of the modified Kneser-Ney algorithm by 40x and reduce its memory usage by 40%.

We have contributed our implementation of this data structure and assorted algorithms to the research community as the MIT Language Modeling (MITLM) toolkit [73], available at `http://www.sls.csail.mit.edu/mitlm/`. We invite the community to not only utilize this toolkit for language modeling techniques previously considered impractical, but also consider implementing future language modeling techniques with MITLM.

The following summarizes the main contributions of this thesis:

- Generalized linear interpolation for context-sensitive combination of language models.
- n-gram weighting to de-emphasize out-of-domain n-grams during LM estimation.
- Unsupervised and active-learning techniques for LM parameter tuning.
- Efficient data structure and algorithms for iterative language model estimation.

1.3 Thesis Overview

We started this thesis by presenting our vision of increasing the diversity and personalization of natural language applications via rapid adaptation. Towards this vision, we will examine various language modeling techniques for domains with limited data in the following chapters:

Chapter 2: Background

We review the basics of n-gram language modeling and introduce the modified Kneser-Ney smoothing algorithm. After describing the lecture transcription task, we summarize related work in the field of language model adaptation.

Chapter 3: Style & Topic Adaptation

We present our earlier work in language model adaptation that motivated many of the approaches proposed in this thesis. Specifically, we will examine the use of the Hidden Markov Model with Latent Dirichlet Allocation (HMM-LDA) [67] for selecting relevant n-grams to train style-specific and topic-specific n-gram component models.

Chapter 4: Generalized Linear Interpolation

We propose a generalization of linear interpolation that computes context-dependent mixture weights from features predictive of component relevance. The resulting model subsumes static linear interpolation [138] and count merging [1], and obtains a statistically significant improvement in both perplexity and word error rate.

Chapter 5: n-gram Weighting

Since the n-grams from partially matched corpora may not be of equal relevance to the target domain, we introduce an n-gram weighting technique to adjust the n-gram counts used during model estimation. We evaluate this approach using a variety of features derived from readily available segmentation and metadata information for each corpus.

Chapter 6: Parameter Optimization

As an in-domain development set may not be available for parameter tuning, we explore the iterative use of the recognition hypotheses for unsupervised parameter estimation. We also evaluate the effectiveness of supervised adaptation using varying amounts of user-provided transcripts of utterances selected via multiple strategies.

Chapter 7: Data Structure & Algorithms

To enable the practical usage of the above techniques, we present an efficient data structure and optimized algorithms specifically designed for iterative parameter tuning. With the resulting implementation, we demonstrate the feasibility and effectiveness of such iterative techniques in language model estimation.

Chapter 8: Conclusion & Future Work

We summarize our contributions, consider possible future work, and discuss the next steps towards the vision that motivated all this work.

Chapter 2

Background

In this chapter, we first review the basics of n-gram language modeling and introduce the modified Kneser-Ney smoothing algorithm. Next, we describe the lecture transcription task along with the data sets and speech recognizer that we consider in this thesis. Finally, we summarize the related work in the field of language model adaptation most relevant to our limited-data scenario.

2.1 Language Models

A statistical language model (LM) assigns a probability to a sequence of words $w_1 \cdots w_m$, drawn from a vocabulary V, corresponding to the likelihood of observing the sequence within an application domain. Language models are used in a wide variety of natural language processing tasks, such as speech recognition [3, 77], handwriting recognition [106], pinyin conversion [47], machine translation [12], and information retrieval [122]. In speech and handwriting recognition, the language model not only serves to disambiguate among acoustically similar phrases such as (FOR, FOUR, FORE) and (RECOGNIZE SPEECH, WRECK A NICE BEACH), but also guides the search for the best acoustically matching word sequence towards ones with the highest language probabilities. The language model can also be used to rank different alternatives in pinyin conversion and machine translation. For information retrieval, modeling the probability that a document generates a given query using language modeling outperforms traditional tf-idf techniques.

One of the main challenges facing language modeling is the issue of out-of-vocabulary (OOV) words. In many tasks, such as speech and handwriting recognition, the set of vocabulary words is not known ahead of time. Since unknown words cannot be hypothesized by the recognizer, they not only contribute to recognition errors directly, but also indirectly by interfering with the recognition of neighboring words. As the vocabulary depends on the topic of interest, effective language models need to adapt to the application domain.

In many speech recognition applications, the language model will be used to describe spontaneous speech containing disfluencies, such as filled pauses (UM, UH) and repairs (FOR W- FOR THIS ENTRY). As most available training data contains text in formal written style, effective language models in these scenarios need to adapt to the casual conversational style.

2.1.1 n-gram Models

An n-gram model is a simple and popular class of language models. Using the chain rule from probability theory, we can decompose the joint probability of a word sequence into a

product of conditional probabilities:

$$p(w_1 \cdots w_m) = p(w_1)p(w_2|w_1)p(w_3|w_1w_2) \cdots p(w_m|w_1 \cdots w_{m-1}) = \prod_{i=1}^{m} p(w_i|w_1 \cdots w_{i-1})$$

Applying the Markov assumption that the conditional probability of each word only depends on the previous $n-1$ words, we approximate the sequence probability as:

$$p(w_1 \cdots w_m) = \prod_{i=1}^{m} p(w_i|w_1 \cdots w_{i-1}) \approx \prod_{i=1}^{m} p(w_i|w_{i-n+1} \cdots w_{i-1}) \equiv \prod_{i=1}^{m} p(w_i|w_{i-n+1}^{i-1})$$

where w_i^j denotes the word sub-sequence $w_i \cdots w_j$. By definition, an n-gram is a sequence of n words, with the terms *unigram*, *bigram*, and *trigram* referring to an n-gram with $n =$ 1, 2, and 3, respectively. Thus, an n-gram model approximates the probability of a word sequence using the conditional probabilities of the embedded n-grams. The order of an n-gram model refers to the value n.

When using an n-gram model to compute the probability of a sentence, we pretend that each sentence is padded with special beginning of sentence tokens <S>, such that $w_{-n+2} = \cdots = w_0 =$ <S>. Similarly, to ensure that the probability of all sentences of varying lengths sum up to 1, we attach a special end of sentence token </S> at the end and include it in the product of the conditional probabilities. For example, we can compute the bigram and trigram probabilities of the sentence HOW ARE YOU as:

$$\begin{aligned} p_{\text{bigram}}(\text{HOW ARE YOU}) &= p(\text{HOW}|\text{<S>})p(\text{ARE}|\text{HOW})p(\text{YOU}|\text{ARE})p(\text{</S>}|\text{YOU}) \\ p_{\text{trigram}}(\text{HOW ARE YOU}) &= p(\text{HOW}|\text{<S> <S>})p(\text{ARE}|\text{<S> HOW}) \cdots p(\text{</S>}|\text{ARE YOU}) \end{aligned}$$

An n-gram model $p(w_i|w_{i-n+1}^{i-1})$ is comprised of a set of discrete distributions over words w_i in the vocabulary V, one distribution for each n-gram history or context $h_i = w_{i-n+1}^{i-1}$. Estimating $p(w_i|w_{i-n+1}^{i-1}) = p(w_i|h_i)$ generally involves counting the fraction of times the target word w_i appears after the history h_i in a training text corpus. If we denote the number of times an n-gram hw appears in the corpus as $c(hw)$, a simple estimate of the n-gram probability is given by:

$$p(w|h) = \frac{c(hw)}{\sum_{w'} c(hw')}$$

This is known as the maximum likelihood (ML) estimate of the n-gram mode, as it maximizes the probability of the training data.

For example, given a training corpus consisting of the following four sentences:

HOW ARE YOU WHERE ARE YOU WHERE SEE YOU LATER

maximum likelihood estimation gives us:

$$p(\text{HOW}|\text{<S>}) = \frac{c(\text{<S> HOW})}{\sum_w c(\text{<S> } w)} = \frac{1}{4}$$

$$p(\text{YOU}|\text{ARE}) = \frac{c(\text{ARE YOU})}{\sum_w c(\text{ARE } w)} = \frac{2}{2}$$

$$
\begin{aligned}
p(\text{</S>}|\text{YOU}) &= \frac{c(\text{YOU </S>})}{\sum_w c(\text{YOU } w)} = \frac{2}{3} \\
p(\text{ARE}|\text{YOU}) &= \frac{c(\text{YOU ARE})}{\sum_w c(\text{YOU } w)} = \frac{0}{3} \\
p(\text{</S>}|\text{ARE YOU}) &= \frac{c(\text{ARE YOU </S>})}{\sum_w c(\text{ARE YOU } w)} = \frac{2}{2} \\
p(\text{<S>}) &= \frac{c(\text{<S>})}{\sum_w c(w)} = \frac{0}{14} \\
p(\text{YOU}) &= \frac{c(\text{YOU})}{\sum_w c(w)} = \frac{3}{14} \\
p(\text{</S>}) &= \frac{c(\text{</S>})}{\sum_w c(w)} = \frac{4}{14}
\end{aligned}
$$

Thus, the bigram probability of the sentence HOW ARE YOU is given by:

$$
\begin{aligned}
p(\text{HOW ARE YOU}) &= p(\text{HOW}|\text{<S>})p(\text{ARE}|\text{HOW})p(\text{YOU}|\text{ARE})p(\text{</S>}|\text{YOU}) \\
&= \frac{1}{4} \times \frac{1}{1} \times \frac{2}{2} \times \frac{2}{3} = \frac{1}{6}
\end{aligned}
$$

To obtain a maximum likelihood estimate, we define $c(\text{<S>}) = 0$ while $c(\text{</S>}) = 4$. Although it is tempting to assume that $c(h) = \sum_w c(hw)$, it is not true for n-grams involving the beginning and end of sentence tokens, as $\sum_w c(\text{<S> } w) = 4 \neq 0$ and $\sum_w c(\text{</S> } w) = 0 \neq 4$. We can eliminate such discrepancies by merging the beginning and end of sentence tokens into a single sentence boundary token. However, for consistency with previous work in language modeling [22, 138], we decided to maintain separate tokens in the presentation of this thesis.

2.1.2 Evaluation Metrics

Perplexity

A common way of evaluating the effectiveness of an n-gram language model is to measure how well it predicts the sentences from a disjoint evaluation or test corpus. Specifically, given a test set T composed of sentences $s^1, \ldots, s^{|T|}$, where $s^t = w_1^t \cdots w_{|s_t|}^t$, we can compute the probability of the test set as the product of the individual sentence probabilities:

$$
p(T) = \prod_{t=1}^{|T|} p(s^t) = \prod_{t=1}^{|T|} \prod_{i=1}^{|s^t|} p(w_i^t | h_i^t) = \prod_{i=1}^{N} p(w_i | h_i)
$$

where we mapped the words w_i^t and histories h_i^t (including the end of sentence token </S>) for all sentences in the test set into a single sequence w_i and h_i with $N = \sum_{t=1}^{|T|} |s^t|$ words.

To normalize against the size of the test set, we use the information theoretic measure of *cross-entropy*, defined in this case as the average number of bits needed to encode each word in a test set T, given the n-gram model $p(w|h)$:

$$
H_p(T) = -\frac{1}{N} \sum_{i=1}^{N} \log_2 p(w_i | h_i)
$$

From the cross-entropy, we can derive the more commonly used measure of *perplexity*, which computes the reciprocal geometric mean of the probability assigned to each word in the test set:

$$\mathrm{PP}_p(T) = 2^{H_p(T)} = \frac{1}{\sqrt[N]{\prod_{i=1}^{N} p(w_i|h_i)}} = \exp\left[-\frac{1}{N}\sum_{i=1}^{N}\log p(w_i|h_i)\right]$$

Perplexity can also be roughly interpreted as the average number of word types following an n-gram context. Lower perplexity values correspond to more predictive models, whereas higher perplexity values indicate more model uncertainty.

If a probability of zero is assigned to any n-gram in the test corpus, the perplexity degenerates to infinity, making comparisons among different models impossible. Such a scenario actually occurs quite often as the test set frequently contains vocabulary not observed in the training data. Thus, in practice, n-grams with zero probabilities are excluded from the perplexity computation. Instead, counts are maintained for zero probability n-grams, categorized by whether the target word is out-of-vocabulary [138].

With appropriate smoothing (see Section 2.1.3), an n-gram with an in-vocabulary target word should never receive zero probability. However, n-grams with out-of-vocabulary target words will always be assigned zero probabilities and excluded from the perplexity computation. Thus, we cannot generally compare the perplexities of n-gram models trained with different vocabularies, as they are computed over different subsets of the test set n-grams.

Word Error Rate

Although well-motivated from an information theoretic perspective, minimizing perplexity is generally not the ultimate objective in most language modeling applications. In speech recognition applications, a more common evaluation metric is the recognition *word error rate* (WER), computed as the number of word errors made divided by the total number of words in the correct reference transcript. Formally, given a reference transcript $r = r_1 \cdots r_R$ and hypothesized text $h = h_1 \cdots h_H$, the number of word errors is defined as the minimum number of word substitutions, insertions, and deletions needed to transform the reference r to the hypothesis h. This is also known as the Levenshtein or edit distance between the two word sequences.

For example, the following lists the number of substitution (S), deletion (D), and insertion (I) errors of various hypotheses, compared with reference transcript HOW ARE YOU:

HOW ARE YOU	(0)	HOW ARE	(1 D)	HOW'RE YOU	(1 S + 1 D)
HAL ARE YOU	(1 S)		(3 D)	HAL ARE YOU U	(1 S + 1 I)
HAL R U	(2 S)	HOW ARE YOU U	(1 I)	ARE YOU U	(1 D + 1 I)

In general, the word error rate is bounded between $\frac{|R-H|}{R}$ and $\frac{\max(R,H)}{R}$. Thus, when there are more words in the hypothesis than the reference, the word error rate can actually exceed 100%. For speech recognition tasks, the word error rate is generally considered to be a more meaningful evaluation metric for language models than perplexity.

When comparing the word error rates between two systems evaluated on the same data, in addition to reporting the absolute difference, we typically also include a measure of statistical significance known as the *p-value*. The *p-value* corresponds to the probability that two models with the same word error rate over the underlying target distribution will exhibit the observed difference due to random sampling. For speech recognition tasks, statistical

significance is generally computed using the NIST Matched-Pair Sentence-Segment Word Error (MAPSSWE) test [51, 118]. Generally, we consider a result statistically significant if $p \leq 0.05$.

2.1.3 Smoothing and Backoff

With a vocabulary of size $|V|$, it takes $|V|-1$ parameters[1] to characterize each of the $|V|^{n-1}$ word distributions, one for each n-gram history, for a total of $|V|^n - |V|^{n-1}$ parameters. Even with a large one billion word corpus and a moderate vocabulary size of 10,000 words, more than 99.9% of all possible trigrams will not be observed. Under maximum likelihood estimation, these trigrams will all be assigned zero probability. For many language modeling applications, such as speech recognition, having zero n-gram probabilities implies that such word sequence can never by hypothesized, regardless of how clearly the user says it. In general, we want to assign non-zero probabilities to all n-grams to avoid such recognition errors.

Various techniques have been developed over the past two decades to smooth the n-gram model such that no n-gram is assigned zero probability [22, 82, 84, 90, 142, 152] (see [22] for a summary with detailed comparisons). Smoothing generally also improves the estimation of the underlying n-gram probabilities, as it distributes probability mass from observed to unseen n-grams.

Uniform Backoff

One technique for smoothing an n-gram model is to subtract a small constant discount $D < 1$ from the count of each observed n-gram and distribute the withheld counts uniformly to unseen n-grams.

$$p_{\text{unif}}(w|h) = \begin{cases} \frac{c(hw)-D}{\sum_{w'} c(hw')} & c(hw) > 0 \text{ (observed)} \\ \alpha_{\text{unif}}(h) & c(hw) = 0 \text{ (unseen)} \end{cases}$$

Following Chen and Goodman [22], we'll define $N_{1+}(h\bullet) = |\{w : c(hw) > 0\}|$ as the number of observed n-grams with history h. Thus, to ensure that $\sum_w p(w|h) = 1$, we set:

$$\alpha_{\text{unif}}(h) = \frac{D}{\sum_{w'} c(hw')} \frac{N_{1+}(h\bullet)}{|V| - N_{1+}(h\bullet)}$$

Proportional Backoff

Distributing the withheld counts uniformly implies that unseen bigrams like <S> YOU and <S> LATER are assigned the same probability. However, given that the unigram YOU appears much more frequently than LATER, its corresponding bigram should be assigned higher weight. Following this intuition, we can divide the withheld counts proportionally to the probability predicted by the lower-order n-gram model. Specifically, the lower-order backoff probability of an n-gram hw is defined as $p(w|h')$, where h' is derived by truncating the most distant (first) word from history h. For example, the n-gram SEE YOU </S> has

[1] There is one fewer parameter than the number of words as $\sum_w p(hw) = 1$.

history SEE YOU and backoff YOU </S>. Thus, we have:

$$p_{\text{prop}}(w|h) = \begin{cases} \frac{c(hw)-D}{\sum_{w'} c(hw')} & \text{if } c(hw) > 0 \text{ (observed)} \\ \alpha_{\text{prop}}(h) p_{\text{prop}}(w|h') & \text{if } c(hw) = 0 \text{ (unseen)} \end{cases}$$

$$\alpha_{\text{prop}}(h) = \frac{1 - \sum_{w:c(hw)>0} p_{\text{prop}}(w|h)}{1 - \sum_{w:c(hw)>0} p_{\text{prop}}(w|h')}$$

where $\alpha_{\text{prob}}(h)$ is computed to ensure a proper distribution. This form of backoff n-gram model uses the lower order $(n-1)$-gram model to estimate the probability of unseen n-grams. Conventionally, the unigram model backs off to a uniform distribution over the vocabulary V, as not all words may be observed. To obtain proper distributions, the vocabulary includes </S>, but not <S>.

Absolute Discounting

With the proportional backoff smoothing, equally observed n-grams with the same history, such as WHERE ARE and WHERE </S>, are assigned the same probability. However, given that </S> appears more frequently than ARE, we may want to bias the probability slightly towards WHERE </S>. One approach is to interpolate all n-gram probabilities with the lower-order backoff model, whether or not the n-grams are observed. In this case, we obtain:

$$p_{\text{abs}}(w|h) = \frac{\max(c(hw) - D, 0)}{\sum_{w'} c(hw')} + \alpha_{\text{abs}}(h) p_{\text{abs}}(w|h') \qquad \alpha_{\text{abs}}(h) = \frac{D \cdot N_{1+}(h\bullet)}{\sum_{w'} c(hw')}$$

This smoothing technique is known as absolute discounting [22,113]. Although the discount parameter D can be tuned to optimized performance on a disjoint development set corpus, it is often estimated using the n-gram count statistics as:

$$D = \frac{n_1}{n_1 + 2n_2}$$

where n_1 and n_2 are the number of n-grams that appear exactly once and twice in the training corpus.

Kneser-Ney

Observing that the lower-order models have a significant contribution to the overall n-gram model only when the higher-order n-grams have low counts, Kneser and Ney [90] modified the counts used to estimate the lower-order models to better reflect these conditions. Specifically, instead of using the original count $c(hw)$, Kneser and Ney proposed using what we term the left-branching count $c^l(hw) = N_{1+}(\bullet hw) = |\{w' : c(w'hw) > 0\}|$, or the number of unique words that precede the n-gram hw, when estimating the lower-order models. See Chen and Goodman [22] for a detailed account on the intuition and derivation behind the use of the left-branching count in Kneser-Ney smoothing.

Modified Kneser-Ney

Up to now, we have assumed a constant discount value for all observed n-grams. However, as empirically shown in Chen and Goodman [22], the ideal discount value actually varies

with the n-gram count. Thus, instead of using a single discount D for all non-zero counts, modified Kneser-Ney smoothing applies three different discount parameters, D_1, D_2, and D_{3+}, for n-grams with 1, 2, and 3 or more counts, respectively:

$$p_{\text{ModKN}}(w|h) = \frac{\max(c(hw) - D(c(hw)), 0)}{\sum_{w'} c(hw')} + \alpha_{\text{ModKN}}(h) p_{\text{ModKN}}(w|h')$$

$$D(c) = \begin{cases} 0 & \text{if } c = 0 \\ D_1 & \text{if } c = 1 \\ D_2 & \text{if } c = 2 \\ D_{3+} & \text{if } c \geq 3 \end{cases} \qquad \alpha_{\text{ModKN}}(h) = \frac{D_1 N_1(h\bullet) + D_2 N_2(h\bullet) + D_{3+} N_{3+}(h\bullet)}{\sum_{w'} c(hw')}$$

The additional discount parameters can be analogously estimated from count statistics.

$$Y = \frac{n_1}{n_1 + 2n_2} \qquad D_1 = 1 - 2Y\frac{n_2}{n_1} \qquad D_2 = 2 - 3Y\frac{n_3}{n_2} \qquad D_{3+} = 3 - 4Y\frac{n_4}{n_3}$$

In general, we can obtain better language modeling performance by tuning the discount parameters on a development set, especially when there is a slight mismatch between the underlying distributions of the training and development data.

Extending this intuition, we can increase the number of discount parameters beyond three. Preliminary experiments suggest that even though extended Kneser-Ney smoothing can further reduce the language model perplexity, it is prone to overfit the development set and the perplexity reductions do not always translate into improved speech recognition performance.

Summary

In this section, we introduced a small subset of n-gram smoothing techniques. Although different smoothing techniques can be applied independently to each n-gram order, a consistent application of modified Kneser-Ney smoothing to all n-gram orders is currently considered the state-of-the-art technique for most language modeling tasks. Thus, in this thesis, we will explore various techniques that build on top of modified Kneser-Ney smoothing to improve language modeling in limited-data domains.

2.2 Language Modeling for Lecture Transcription

With the increasing availability of audio-visual material over the web, accurate transcripts are needed to improve the search, navigation, summarization, and even translation of the content [55, 62, 158]. Unfortunately manual transcription of the audio is time consuming, expensive, and error-prone. Professional transcription services often charge more than $100 per hour of audio and take up to 3-5 business days. Furthermore, the resulting transcript may still contain significant amount of errors, especially for presentations with highly technical terminologies such as academic lectures. For example, the transcript of a lecture on speech recognition is likely to contain the following errors: *Markov* → *mark of*, *Fourier* → *for your*. As a result of these challenges, there is a significant interest in using speech recognition technology to provide automatic transcriptions. Towards this goal, we will focus in this work on improving the language model for academic lecture transcription [45, 56].

Compared with other types of audio data, lecture speech often exhibits a high degree of spontaneity. Unlike prepared speech, spontaneous speech often contains significant disfluen-

Dataset	Vocab Size	# Words	# Sentences	# Docs
CS-Dev	3,286	97,479	4,126	10
CS-Test	3,354	91,138	3,611	10
Textbook	4,683	138,042	6,762	271
Lectures	29,710	2,123,120	128,895	230
Switchboard	26,834	3,392,571	262,744	4,876

Table 2.1: Summary of the evaluation datasets. Counts include </s>, but not <s>.

cies (repetition, hesitation, rephrasing, partial words) [56, 102]. Furthermore, the speaking rate is generally faster than read speech with large variations across the same session [112]. Thus, existing techniques for transcribing read speech often perform poorly when applied to lecture speech.

Unlike general conversational speech that also exhibit a spontaneous style, lecture speech often focuses on narrow topics with special terminologies. Although an hour-long lecture might only contain 800 unique words, 20% of them do not appear among the top 1000 words from general conversational speech or broadcast news transcripts [56]. Even with a 10,000 word vocabulary, 6% are still out-of-vocabulary.

In this thesis, we propose various language modeling techniques for domains with limited data and evaluate them on a lecture transcription task. The following sections describe the evaluation task and speech recognition system in more detail.

2.2.1 Evaluation Task

We evaluate the perplexity and word error rate of various n-gram language models on 20 hour-long lectures drawn from an introductory computer science course. For most experiments, we withhold the first 10 lectures as the development set (CS-Dev) and use the last 10 as the test set (CS-Test).

As the computer science lectures exhibit both the spontaneous lecture style and a topic-specific vocabulary, we would ideally train the language model using previously transcribed data that matches both the style and topic of these lectures. However, as such data is generally unavailable, we will instead train the language model from the available partially matched data, including the content of the course textbook and the transcripts of general lectures and spoken conversational speech. Table 2.1 summarizes all the data used in the evaluation, while Figure 2-1 contains excerpts from these corpora.

Course Textbook

The course textbook[2] for the introductory computer science class provides an excellent source of material for the specialized terminologies on the target topic. As with most textbooks, it is hierarchically subdivided into chapters and sections, and written in a formal style with capitalization and punctuations. In our specific case, the textbook has been converted to an HTML format, although in general, it may be in LaTeX, Adobe PDF, Microsoft Word, or other document formats.

To construct the Textbook data set, we first extracted the text from the HTML documents. After breaking the text into sentences at sentence-ending punctuations, we removed

[2] `http://mitpress.mit.edu/sicp/full-text/book/book.html`

CS-Dev **and** CS-Text

now i'm also going to write down another simple simple one just like this which is the mathematical expression the sum of the squares from i equals a to b
<cough> and again very simple program <mumble> and indeed it starts the same way
if a is greater than b then the answer is zero
and of course we're beginning to see that there is something wrong with me writing this down again
okay it's the same program
it's the sum of the square of a and the sum of the squares of the increments and b
now if you look at these things these programs are almost identical
right there's not there's not much to distinguish them

Textbook

this gives the same answer as our previous evaluation model but the process is different
in particular the evaluations of and are each performed twice here corresponding to the reduction of the expression
with x replaced respectively by and
this alternative fully expand and then reduce evaluation method is known as normal order evaluation in contrast to the evaluate the arguments and then apply method that the interpreter actually uses which is called applicative order evaluation
it can be shown that for procedure applications that can be modeled using substitution and that yield legitimate values normal order and applicative order evaluation produce the same value
lisp uses applicative order evaluation partly because of the additional efficiency obtained from avoiding multiple evaluations of expressions such as those illustrated with and above and more . . .

Lectures

okay let's get started now i'm assuming i'm assuming that <um> a you went to recitation yesterday
<uh> b that even if you didn't that
<uh> you know how to separate variables and you know how to construct simple models
of solve physical problems with differential equations and possibly even solve them
so you should have learned that either in high school or eighteen oh one <uh> here
or
<uh>
yeah
so i'm going to start from that point i assume you know that i'm not going to tell you what differential equations are or what modeling is if you

Switchboard

<noise>
<laugh> lost a lot of hair over that project <uh> what we run into <um> is we have the texas air control board t a c b that send out <uh> jurisdictions under which we have to <uh> <uh> reply to
and a lot of their rules and regulations aren't real clear so we have our manager of environmental who assist the t a c b which is located in austin in writing and hey look what we've done here at t i
as well as what society can do to improve the air quality of the atmosphere around us
and <uh> we are presently <uh> in receipt of a site permit which will allow us to <um>
<uh> this is air side <partial> allow to have certain emissions up to a certain tonnage it's in in in tons per year <um>

Figure 2-1: Excerpts from the evaluation corpora.

all punctuations and capitalizations as they generally are not explicitly spoken in speech. Since some words, such as numbers, abbreviations, and acronyms, are not pronounced as written, we heuristically normalized these words into their corresponding spoken forms (for a more principled approach to text normalization, see Sproat et al. [136]). Any content that cannot be automatically processed, such as figures and algebraic expressions, is removed. The resulting corpus consists of approximately 138k words, split across 271 documents, one for each section of the textbook.

General Lectures

As a corpus with the desired spontaneous lecture style, the Lectures data set consists of transcripts from a variety of general seminars and academic lectures covering a wide range of topics [56]. Each of the 230 lectures is annotated by course (6.001, MIT-World-2004, ...) and speaker (WalterLewin, GilbertStrang, ...). Noises in the audio are categorized into <BACKGROUND>, <COUGH>, <CROSSTALK>, <LAUGH>, <MUMBLE>, or <NOISE>, whereas filled pauses are transcribed as <EH>, <ER>, <UH>, or <UM>. Furthermore, partially spoken words are replaced with <PARTIAL>, while words that cannot be understood are labeled as <UNINTELLIGIBLE> or <UNKNOWN>.

Switchboard

Although not as well-matched to the target lecture style as the Lectures corpus, the LDC Switchboard corpus of spontaneous conversational speech [58] does provide significantly more data for language model training. Although previous research generally showed that more data leads to better models, they assumed that the data matches the target domain. To evaluate the effectiveness of using more, but only partially matched, data for language model estimation, we will consider the Switchboard corpus as a third training set.

2.2.2 Speech Recognizer

For all speech recognition experiments in this work, we employ the landmark-based SUMMIT speech recognition system developed in the Spoken Language Systems group at MIT [54]. The acoustic front-end consists of standard Mel-frequency cepstral coefficients with cepstral mean normalization [77]. The acoustic model features are computed from hypothesized phonetic landmarks and whitened using principal component analysis [54]. Speaker independent acoustic modeling is performed using hierarchical Gaussian mixture models constructed via large-margin discriminative training [18].

Trigram language models are trained with the default training set vocabulary via the various techniques presented in this thesis. Each resulting model is composed with the context-dependent phone model, phonological model, and pronunciation dictionary to form the finite state transducer used to decode each input spoken utterance [109, 162].

In our recognition experiments, the audio is first pre-segmented into utterances via forced alignment against the reference transcript [68]. We next process each utterance using the SUMMIT speech recognizer and generate a hypothesis and optionally a word lattice. As we are primarily interested in the recognition performance of the content words, we first remove all noise and disfluency tags (cf. Section 2.2.1) from both the reference and hypothesized transcripts before computing the word error rate.

As all language modeling techniques presented in this work yield backoff n-gram models, they can be applied directly during the first recognition pass instead of through a subsequent

n-best or lattice rescoring step. However, for efficient performance evaluation of language model variations, some of the reported results, when specified, are computed via lattice rescoring. Preliminary experiments suggest that lattice rescoring incurs a search error loss of less than 0.1% word error rate, generally not statistically significant for this particular task.

2.3 Related Work

A significant amount of previous research has investigated language modeling for lecture transcription [17, 44, 55, 56, 83, 97, 112, 115, 116, 119, 151, 155], generally employing various data collection, text selection, and model interpolation techniques. In broader contexts, using partially matched data for language model training belongs to a class of algorithms known as language model adaptation (see Bellegarda [6] for a review). In the following sections, we will summarize some of the most relevant existing work in language model adaptation and describe how they relate to our scenario.

2.3.1 Data Collection

The first step in language model training is to collect data that matches the target domain as closely as possible. In domains like broadcast news, a large collection of news transcripts may already be available [63]. For other applications, relevant data may be found from a diverse variety of sources, such as query logs for voice search [43] and lecture slides for lecture transcription [119].

Recently, there has been a growing interest in gathering related data from the World Wide Web [14, 15, 95, 110, 131, 160]. Typically, keyword queries constructed from in-domain data are fed to search engines to find web documents related to the domain topic. However, since not all results, or even sections of the relevant documents, match the target domain, various text selection and filtering approaches have been proposed to extract a subset of the web corpus that best fit the target topic and style [14, 111, 115, 132, 133].

Ultimately, the best matched data for speech recognition language modeling is the test data itself, precisely the untranscribed data that we are trying to recognize. Although the reference transcript is not available for training, we can utilize the hypotheses from the previous recognition pass to perform unsupervised self-adaptation and iterative recognition [1, 2, 20, 65, 115, 135, 144]. To avoid reinforcing recognition errors through self-training, previous work has also explored the use of confidence at the utterance and word level to weight and filter the recognition hypotheses [65, 135].

Although we do not directly investigate approaches to collect relevant data in this thesis, it is a crucial first step to language model adaptation and is complementary to the techniques presented in this work. Despite techniques for selecting a subset of the collected data to improve its similarity to the domain of interest, the resulting text still often exhibit some degree of mismatch, providing opportunities for additional n-gram level filtering that we will explore in this work.

2.3.2 Interpolation

The typical scenario explored in previous language model adaptation research involves adapting a single background model $p_B(w|h)$ trained from a large general corpus with a domain-specific model $p_A(w|h)$ built from the in-domain adaptation corpus. In this setting,

the easiest and most popular technique is to combine the two models via linear interpolation (LI) [81], where we take the weighted average of the probabilities assigned by the individual LMs as the adapted probability. Specifically,

$$p^{\text{LI}}_{\text{dynamic}}(w|h) = (1-\lambda)p_B(w|h) + \lambda p_A(w|h)$$

where $0 \leq \lambda \leq 1$ is the interpolation weight. The interpolation weight can be selected empirically, or tuned to minimize the perplexity of a disjoint development set via the Expectation-Maximization (EM) algorithm [33, 130].

To combine more than two language models trained from disparate sources, we can extend linear interpolation to a mixture of components $p_i(w|h)$ as follows:

$$p^{\text{LI}}_{\text{dynamic}}(w|h) = \sum_i \lambda_i p_i(w|h) \qquad \lambda_i \geq 0 \qquad \sum_i \lambda_i = 1$$

where λ_i are non-negative interpolation weights that sum to 1. With mixture models, we can combine models that partially match the target domain in topic and style, not necessarily requiring an in-domain model [50, 119].

In practice, a mixture model is less efficient for speech recognition than a single n-gram backoff model, as it involves multiple probability evaluations for each possible word expansion and requires more storage for the multiple component models. Thus, as an approximation [138], we will construct a static n-gram model where the probability of an observed n-gram is given by the weighted average of the component model probabilities. The remaining probabilities are computed via appropriately normalized backoff probabilities. Specifically,

$$p^{\text{LI}}(w|h) = \begin{cases} \sum_i \lambda_i p_i(w|h) & \text{if } \sum_i c_i(hw) > 0 \\ \alpha(h) p^{\text{LI}}(w|h') & \text{otherwise} \end{cases}$$

where $c_i(hw)$ is the observation count of the n-gram hw for model i. Empirically, the resulting static linear interpolation generally achieves lower perplexity than the original model [138]. Thus, in subsequent usages, we will refer to the static version as linear interpolation and label the original version as dynamic linear interpolation.

As a variant of linear interpolation, log-linear interpolation (LLI) [89] combines models by interpolating the n-gram probabilities in the log domain, giving us:

$$p^{\text{LLI}}(w|h) = \frac{1}{Z_\theta(h)} \prod_i p_i(w|h)^{\theta_i}$$

where θ_i are unconstrained log-linear interpolation weights and $Z_\theta(h) = \sum_{w'} \prod_i p_i(w'|h)^{\theta_i}$ is a history-dependent normalization factor. Although demonstrated to be more effective than linear interpolation, log-linear interpolation is not in wide use as tuning the weights involves numerical optimization and is more computationally expensive than simple linear interpolation. Furthermore, the resulting model cannot be represented as a standard n-gram backoff model, reducing the applicability of the model.

Instead of interpolating the component model probabilities, we can also merge the models at the count level, resulting in a technique multiply known as count merging, count mixing, and MAP adaptation [1, 2, 21, 38, 70, 107]. To combine two models using count

merging with n-gram smoothing and backoff [1], we use:

$$p^{\text{CM}}_{\text{dynamic}}(w|h) = \frac{c^{\text{disc}}_B(hw) + \varepsilon c^{\text{disc}}_A(hw)}{\sum_w c_B(hw) + \varepsilon \sum_w c_A(hw)}$$

$$p^{\text{CM}}(w|h) = \begin{cases} p^{\text{CM}}_{\text{dynamic}}(w|h) & \text{if } c_B(hw) + c_A(hw) > 0 \\ \alpha(h) p^{\text{CM}}(w|h') & \text{otherwise} \end{cases}$$

where $c_B(hw)$ and $c_A(hw)$ are the observation counts of the n-gram hw in the background and adaptation corpora, respectively; $c^{\text{disc}}_B(hw)$ and $c^{\text{disc}}_A(hw)$ are the corresponding discounted counts after smoothing; ε is the count merging parameter; and $\alpha(h)$ is the history dependent backoff weight computed to ensure that $\sum_w p^{\text{CM}}(w|h) = 1$. Although count merging generally performs better than linear interpolation without increasing the complexity of the resulting model, tuning the parameters in the past has typically involved empirical trial and error techniques and the process has been described as difficult to optimize [50].

Interpolation-based techniques are the most popular approach to building language models when given multiple partially matched data sources. However, with few exceptions [21], the interpolation parameters are generally fixed across n-gram contexts. When combining data that match the target domain in distinct ways, such a strategy may not be the most optimal. Furthermore, as parameter tuning remains a key challenge to the adoption of better performing interpolation techniques, a set of efficient and automatic tools for language model estimation and parameter optimization has the potential to propel language modeling research forward.

2.3.3 MDI Adaptation

Instead of interpolating n-gram probabilities or counts, another branch of language model adaptation research examines models based on minimum discrimination information (MDI) [121, 129]. In this maximum entropy inspired technique, we select the n-gram distribution with the closest Kullback-Leibler divergence to the background model that satisfies a set of marginal constraints as the adapted model. Such a distribution has traditionally been estimated using the generalized iterative scaling algorithm [29]. However, experiments suggest that limited-memory quasi-Newton numerical optimization techniques, such as L-BFGS [159], achieve significantly faster convergence [104].

Previous works have explored using n-gram probabilities estimated from the adaptation data as the MDI constraints [125,126,129]. For unigram constraints, a closed form solution is available [39,91]. To better estimate the constraints given limited adaptation data, topic modeling techniques have been used to smooth the unigram probabilities over topically similar words [40, 141]. In general, MDI adaptation allows arbitrary constraints to be specified, including long distance syntactic and topic coherence constraints [88,129].

As the MDI adapted models generally cannot be represented as backoff n-gram models, in speech recognition tasks, they are typically applied via a secondary n-best reranking pass [88,129]. To improve the robustness of the estimated parameters, recent research has investigated various smoothing and regularization techniques [23,36,37,59,86]. Nevertheless, as MDI adaptation generally requires a moderate amount of in-domain data in order to estimate effective marginal constraints, it is not suitable for the limited-data domain scenario studied in this work. It is also unclear how MDI adaptation can take advantage of multiple background models that match the target domain in complementary ways.

2.3.4 Dynamic Adaptation

Oftentimes, even when the general topic of the target domain is known, the specific topic of the target text might evolve over time. Thus, instead of estimating a static language model, approaches that dynamically adapt the model based on previously observed content may achieve better performance.

As words observed earlier in a document are likely to appear again, we can adapt the baseline model with a cache n-gram model trained from recently observed text using dynamic linear interpolation [25, 93]. Extending this intuition, we can increase the probability of unobserved but topically related words. Specifically, given a mixture model with topic-specific components [5, 8, 69, 101], we can increase the mixture weights of the topics corresponding to previously observed words to better predict the next word [25, 34, 49, 79]. Similar concept have also been applied to maximum entropy language models [96, 129].

For some adaptation techniques, continually adapting the model may be too costly to be practical. Instead, a multipass technique can be employed where the recognition hypotheses from the first pass is used to adapt the model employed in the second pass [79, 97, 101, 115, 134, 141]. The technique can be repeated iteratively to take advantage of the improved recognition hypothesis [52, 115]. In such a configuration, the recognition hypotheses can not only be used to adapt the model parameters, but also serve as additional training material via unsupervised self-adaptation (cf. Section 2.3.1).

The main challenge with dynamic adaptation techniques that take advantage of the recognition hypotheses is that the effectiveness of model adaptation diminishes with increasing recognition errors. In fact, adapting on the automatic transcript can sometime degrade performance [91]. Thus, recent works have investigated the use of topic modeling and word-level confidence to reduce the effect of recognition errors [1, 65, 141]. In this work, we will build upon this foundation and apply concepts from dynamic adaptation to more sophisticated language modeling techniques.

2.3.5 Domain Adaptation

More broadly, language model adaptation belongs to a class of machine learning problems known as domain adaptation or transfer learning. In addition to language model adaptation, there is growing interest in applying domain adaptation techniques to other natural language processing tasks, such as part-of-speech tagging, named-entity recognition, and automatic capitalization [9, 19, 30, 31].

One approach is to use the model parameters trained from out-of-domain data as regularization priors for a maximum entropy model trained on in-domain data [19]. As this technique reduces to MDI adaptation under certain conditions, it shares many of the same drawbacks when training language models from multiple partially matched corpora.

Another domain adaptation technique models the partially matched data as a mixture of a truly out-of-domain distribution and a shared general distribution [31]. By treating the source of each data instance as a hidden variable, we can improve the estimation of the target model by using only the data from the shared general distribution. Conceptually, this approach is similar to n-gram weighting (see Chapter 5) where we de-emphasize the influence of out-of-domain n-grams. However, unlike n-gram weighting, this approach completely ignores out-of-domain data, instead of tuning its influence on a development set.

Although domain adaptation techniques show great promise to leverage related out-of-domain data to improve the in-domain model, most recent work focuses only on classification

tasks and is thus not directly relevant. For language model adaptation, an effective approach needs to not only incorporate the information from the partially matched corpora in a principled manner, but also take into consideration count smoothing, backoff probability computation, and model interpolation issues specific to n-gram language models. In this thesis, we will investigate a few such techniques.

Chapter 3

Style & Topic Adaptation

3.1 Motivation

Building language models in domains with limited matched data requires taking advantage of the available partially matched data. For lecture transcription, such data may include generic lecture transcripts and the course textbook. As not all n-grams from these partially matched corpora are of equal relevance to the target domain, we would ideally like to identify the out-of-domain n-grams and de-emphasize their contributions when estimating the component n-gram models. Since certain component models are more relevant in particular n-gram contexts, context-dependent interpolation weights should be computed as a function of the relevance of each language model component for each n-gram history. Finally, as the precise topic of the target document is often unknown a priori and may even shift over time, dynamically adapting the language model (LM) over the course of the lecture may prove to be extremely useful for both increasing transcription accuracy, as well as providing evidence for lecture segmentation and information retrieval.

In this chapter, we investigate the application of the syntactic state and semantic topic assignments from the Hidden Markov Model with Latent Dirichlet Allocation (HMM-LDA) [67] to the problem of language modeling. We explore the use of these context-dependent labels to identify the corpus style and learn document-specific topics from both a large number of spoken lectures as well as written text. By dynamically interpolating the lecture style model with topic-specific models, we obtain language models that better describe the subtopic structure within a lecture. Initial experiments demonstrate a 16.1% and 1.1% reduction in perplexity and absolute word error rate (WER), respectively, over a linearly interpolated trigram baseline.

In the following sections, we first summarize related research on adaptive and topic-mixture language models. We then present the HMM-LDA model and examine its ability to learn syntactic classes as well as topics from the textbook and lecture transcripts. Next, we describe a variety of language modeling experiments to combine the style and topic models that we constructed from the state and topic labels with conventional trigram models trained from both spoken and written materials. Furthermore, we demonstrate the use of the combined model in an online adaptive mode. Finally, we summarize the encouraging results from these preliminary experiments that motivated some of the more principled language modeling techniques introduced in subsequent chapters.

3.2 Related Work

The concepts of adaptive and topic-mixture language models have been previously explored by many researchers. Adaptive language modeling exploits the property that words appearing earlier in a document are likely to appear again. Cache language models [25,93] leverage this observation and increase the probability of previously observed words in a document when predicting the next word. By interpolating with a conditional trigram cache model, Goodman [60] demonstrated up to 34% decrease in perplexity over a trigram baseline for small training sets.

The cache intuition has been extended by attempting to increase the probability of unobserved but topically related words. Specifically, given a mixture model with topic-specific components, we can increase the mixture weights of the topics corresponding to previously observed words to better predict the next word. Some of the early work in this area used a maximum entropy language model framework to trigger increases in the likelihood of related words [96, 129].

A variety of methods has been used to explore topic-mixture models. To model a mixture of topics within a document, the sentence mixture model [79] builds multiple topic models from clusters of training sentences and defines the probability of a target sentence as a weighted combination of its probability under each topic model. Latent Semantic Analysis (LSA) has been used to cluster topically related words and has demonstrated significant reduction in perplexity and word error rate [5]. Probabilistic LSA (PLSA) has been used to decompose documents into component word distributions and create unigram topic models from these distributions. Dynamic combination of these unigram topic models with a generic trigram model has been demonstrated to yield noticeable perplexity reduction [49].

To identify topics from an unlabeled corpus, Blei et al. [8] extend PLSA with the Latent Dirichlet Allocation (LDA) model that describes each document in a corpus as generated from a mixture of topics, each characterized by a word unigram distribution. A Hidden Markov Model with LDA (HMM-LDA) [67] further extends this topic mixture model to separate syntactic words from content words, whose distributions depend primarily on local context and document topic, respectively.

In the specific area of lecture processing, previous work in language model adaptation has primarily focused on customizing a fixed n-gram language model for each lecture by combining n-gram statistics from general conversational speech, other lectures, textbooks, and other resources related to the target lecture [97, 111, 112, 119].

Most of the previous work on model adaptation focuses on in-domain adaptation using large amounts of matched training data. However, most if not all of the data available to train a lecture language model are either cross-domain or cross-style. Although adaptive models have been shown to yield significant perplexity reduction on clean transcripts, the improvements tend to diminish when working with speech recognizer hypotheses with a high WER.

In this chapter, we apply the concept of dynamic topic adaptation to the lecture transcription task. Unlike previous work, we first construct a style model and a topic-domain model using the classification of word instances into syntactic states and topics provided by HMM-LDA. Furthermore, we leverage the context-dependent labels to extend topic models from unigrams to n-grams, allowing for better prediction of transitions involving topic words. Note that although this chapter focuses on the use of HMM-LDA to generate the state and topic labels, any method that yields such labels suffices for our experiments. The following section describes the HMM-LDA framework in more detail.

3.3 Hidden Markov Model with Latent Dirichlet Allocation

3.3.1 Latent Dirichlet Allocation

Discrete Principal Component Analysis describes a family of models that decompose a set of feature vectors into its principal components [16]. Describing feature vectors via their components reduces the number of parameters required to model the data, hence improving the quality of the estimated parameters when given limited training data. LSA, PLSA, and LDA are all examples from this family.

Given a predefined number of desired components, LSA models feature vectors by finding a set of orthonormal components that maximize the variance using singular value decomposition [32]. Unfortunately, the component vectors may contain non-interpretable negative values when working with word occurrence counts as feature vectors. PLSA eliminates this problem by using non-negative matrix factorization to model each document as a weighted combination of a set of non-negative feature vectors [69]. However, because the number of parameters grows linearly with the number of documents, the model is prone to overfitting. Furthermore, because each training document has its own set of topic weight parameters, PLSA does not provide a generative framework for describing the probability of an unseen document [8].

To address the shortcomings of PLSA, Blei et al. [8] introduced the LDA model, which further imposes a Dirichlet distribution on the topic mixture weights corresponding to the documents in the corpus. With the number of model parameters dependent only on the number of topic mixtures and vocabulary size, LDA is less prone to overfitting and is capable of estimating the probability of unobserved test documents.

Empirically, LDA has been shown to outperform PLSA in corpus perplexity, collaborative filtering, and text classification experiments [8]. Various extensions to the basic LDA model have since been proposed. The Author Topic model adds an additional dependency on the author(s) to the topic mixture weights of each document [128]. The Hierarchical Dirichlet Process is a non-parametric model that generalizes distribution parameter modeling to multiple levels. Without having to estimate the number of mixture components, this model has been shown to match the best result from LDA on a document modeling task [143].

3.3.2 HMM-LDA Model

The HMM-LDA model proposed by Griffiths et al. [67] combines the HMM [124] and LDA models to separate syntactic words with local dependencies from topic-dependent content words without requiring any labeled data. Similar to HMM-based part-of-speech taggers [94], HMM-LDA maps each word in the document to a hidden syntactic state. Each state generates words according to a unigram distribution except for the special topic state, where words are modeled by document-specific mixtures of topic distributions, as in LDA. Figure 3-1 describes this generative process in more detail.

Unlike vocabulary selection techniques that separate domain-independent words from topic-specific keywords using word collocation statistics, HMM-LDA classifies each word instance according to its context. Thus, an instance of the word RETURN may be assigned to a syntactic state in TO RETURN A, but classified as a topic keyword in EXPECTED RETURN FOR. By labeling each word in the training set with its syntactic state and mixture topic, HMM-LDA not only separates stylistic words from content words in a context-dependent manner, but also decomposes the corpus into a set of topic word distributions. This form of

For each document d in the corpus:

1. Draw topic weights θ^d from Dirichlet(α).
2. For each word w_i in document d
   (a) Draw topic z_i from Multinomial(θ^d)
   (b) Draw state s_i from Multinomial($\pi^{s_{i-1}}$)
   (c) Draw word w_i from:
   $$\begin{cases} \text{Multinomial}(\beta^{z_i}) & \text{if } s_i = s_{\text{topic}} \\ \text{Multinomial}(\gamma^{s_i}) & \text{otherwise} \end{cases}$$

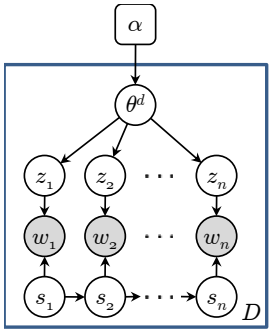


Figure 3-1: Generative framework and simplified graphical model representation of HMM-LDA. The number of states and topics are pre-specified. The topic mixture for each document is modeled with a Dirichlet distribution. Each word w_i in the n-word document is generated from its hidden state s_i or hidden topic z_i if s_i is the special topic state s_{topic}.

soft, context-dependent classification has many potential uses in language modeling, topic segmentation, and indexing.

3.3.3 Training

In this preliminary study, we employ the MATLAB Topic Modeling Toolbox [66,67] to train the HMM-LDA model. This particular implementation performs Gibbs sampling, a form of Markov chain Monte Carlo (MCMC), to estimate the optimal model parameters fitted to the training data. Specifically, the algorithm creates a Markov chain whose stationary distribution matches the expected distribution of the state and topic labels for each word in the training corpus. Starting from random labels, Gibbs sampling sequentially samples the label for each hidden variable conditioned on the current value of all other variables. After a sufficient number of iterations, the Markov chain converges to the stationary distribution. We can easily compute the posterior word distribution for each state and topic from a single sample by averaging over the label counts and prior parameters. With a sufficiently large training set, we will have enough words assigned to each state and topic to yield a reasonable approximation to the underlying distribution.

In the following sections, we examine the application of models derived from the HMM-LDA labels to the task of spoken lecture transcription and explore adaptive topic modeling techniques to construct a better lecture language model.

3.4 Analysis

In the following analysis, we applied HMM-LDA to a subset of the Lectures dataset (cf. Section 2.2.1), consisting of 150 lectures from math, physics, and general seminars. We ran the Gibbs sampler against the reduced Lectures* dataset for a total of 2800 iterations, computing a model every 10 iterations, and took the model with the lowest perplexity as the final model. We chose to build the model with 20 states and 100 topics based on results from preliminary experiments. We also trained an HMM-LDA model on the Textbook dataset using the same model parameters. We ran the sampler for a total of 2000 iterations, computing the perplexity every 100 iterations. Again, we selected the lowest perplexity model as the final model.

1	2	3	4	5
CENTER	WORK	RIGHTS	SYSTEM	<LAUGH>
WORLD	RESEARCH	HUMAN	THINGS	HER
AND	RIGHT	U.	ROBOT	CHILDREN
IDEAS	PEOPLE	S.	SYSTEMS	BOOK
NEW	COMPUTING	GOVERNMENT	WORK	CAMBRIDGE
TECHNOLOGY	NETWORK	INTERNATIONAL	EXAMPLE	BOOKS
INNOVATION	SYSTEM	COUNTRIES	PERSON	STREET
COMMUNITY	INFORMATION	PRESIDENT	ROBOTS	CITY
PLACE	SOFTWARE	WORLD	LEARNING	LIBRARY
BUILDING	COMPUTERS	SUPPORT	MACHINE	BROTHER
6	**7**	**8**	**9**	**10**
<PARTIAL>	CLASS	BASIS	MAGNETIC	LIGHT
MEMORY	PEOPLE	V	CURRENT	RED
AH	TAX	<EH>	FIELD	WATER
BRAIN	WEALTH	VECTOR	LOOP	COLORS
ANIMAL	SOCIAL	MATRIX	SURFACE	WHITE
OKAY	AMERICAN	TRANSFORMATION	DIRECTION	ANGLE
EYE	POWER	LINEAR	E	BLUE
SYNAPTIC	WORLD	EIGHT	LAW	HERE
RECEPTORS	<UNINTELLIGIBLE>	OUTPUT	FLUX	RAINBOW
MOUSE	SOCIETY	T	M	SUN

Table 3.1: The top 10 words from 10 randomly selected topics from the Lectures* dataset.

3.4.1 Semantic Topics

HMM-LDA extracts words whose distributions vary across documents and clusters them into a set of components. In Table 3.1, we list the top 10 words from a random selection of 10 topics computed from the Lectures* dataset. As shown, the words assigned to the LDA topic state are representative of content words and are grouped into broad semantic topics. For example, topic 4, 8, and 9 correspond to machine learning, linear algebra, and magnetism, respectively.

Since the Lectures* dataset consists of speech transcripts with disfluencies, it is interesting to observe that <LAUGH> is the top word in a topic corresponding to childhood memories. Cursory examination of the data suggests that the speakers talking about children tend to laugh more during the lecture. Although it may not be desirable to capture speaker idiosyncrasies in the topic mixtures, HMM-LDA has clearly demonstrated its ability to capture distinctive semantic topics in a corpus. By leveraging all documents in the corpus, the model yields smoother topic word distributions that are less vulnerable to overfitting.

Since HMM-LDA labels the state and topic of each word in the training corpus, we can also visualize the results by color-coding the words by their topic assignments. Figure 3-2 shows a color-coded excerpt from a topically coherent paragraph in the Textbook dataset. Notice how most of the content words (in italics) are assigned the same topic/color. Furthermore, of the 7 instances of the words AND and OR (boxed), 6 are correctly classified as syntactic or topic words, demonstrating the context-dependent labeling capabilities of the HMM-LDA model. Moreover, from these labels, we can identify multi-word topic phrases (e.g. OUTPUT SIGNALS, INPUT SIGNAL, AND GATE) in addition to stand-alone keywords, an observation we will leverage later on with n-gram topic models.

We draw an *inverter symbolically* as in Figure 3.24. An *and gate*, also shown in Figure 3.24, is a *primitive function* box with two *inputs* and *one output*. It drives its *output signal* to a value that is the *logical and* of the *inputs*. That is, if both of its *input signals become 1*. Then *one* and *gate delay* time later the *and gate* will force its *output signal* to be *1*; otherwise the *output* will be *0*. An *or gate* is a *similar two input primitive function* box that drives its *output signal* to a value that is the *logical or* of the *inputs*. That is, the *output* will *become 1* if at least *one* of the *input signals* is *1*; otherwise the *output* will *become 0*.

Figure 3-2: Color-coded excerpt from the Textbook dataset showing the context-dependent topic labels. Syntactic words appear black. Topic words are shown in *italics* with their respective topic colors. All instances of the words AND and OR are boxed.

3.4.2 Syntactic States

Since the syntactic states are shared across all documents, we expect words associated with the syntactic states when applying HMM-LDA to the Lectures* dataset to reflect the lecture style vocabulary.

In Table 3.2, we list the top 10 words from each of the 19 syntactic states (state 1 is the topic state). Note that each state plays a clear syntactic role. For example, state 13 contains prepositions while state 14 contains verbs. Since the model is trained on transcriptions of spontaneous speech, hesitation disfluencies (<UH>, <UM>, <PARTIAL>) are all grouped in state 20 along with other words (SO, IF, OKAY) that frequently indicate hesitation. While many of these hesitation words are conjunctions, the words in state 9 show that most conjunctions are actually assigned to a different state representing different syntactic behavior from hesitations. As demonstrated with spontaneous speech, HMM-LDA yields syntactic states that have a good correspondence to part-of-speech labels, without requiring any labeled training data.

3.4.3 Discussions

Although MCMC techniques converge to the global stationary distribution, we cannot guarantee convergence from observations of the perplexity alone. Unlike EM algorithms, random sampling may actually temporarily decrease the model likelihood. Thus, in the above analysis, the number of iterations was chosen to be at least double the point at which the perplexity first appears to converge.

In addition to the number of iterations, the choice in the number of states and topics, as well as the values of the Dirichlet prior hyperparameters, also impact the quality and effectiveness of the resulting model. Ideally, we run the algorithm with different combinations of the parameter values and perform model selection to choose the model with the best complexity-penalized likelihood. However, given finite computing resources, this approach is often impractical. In our implementation of the HMM-LDA algorithm used in subsequent chapters, we sample over the hyperparameters using a log-normal Metropolis proposal [7]. For future work, we would like to apply the Dirichlet process [143] to automatically estimate the number of states and topics.

Despite the suboptimal choice of parameters and potential lack of convergence, the labels derived from the HMM-LDA model are still effective for language modeling applications, as described next.

1	2	3	4	5	6	7	8	9	10
Topic	KNOW	IT	IT	CAN	IT'S	GOING	A	AND	TO
	SEE	YOU	THIS	WILL	NOT	DOING	AN	BUT	JUST
	DO	OUT	THAT	WOULD	THAT'S	ONE	SOME	OR	LONGER
	THINK	UP	THERE	DON'T	I'M	LOOKING	ONE	BECAUSE	DOESN'T
	GO	THEM	WHICH	COULD	JUST	SORT	NO	AS	NEVER
	GET	THAT	HE	DO	THERE'S	DONE	IN	THAT	GO
	SAY	ME	HERE	JUST	<UH>	ABLE	TWO	WHERE	PHYSICALLY
	MAKE	ABOUT	COURSE	ME	WE'RE	COMING	ANY	THANK	THAT'LL
	LOOK	HERE	WHO	SHOULD	ALSO	TALKING	THIS	WHICH	ANYBODY'S
	TAKE	ALL	THEY	MAY	YOU'RE	TRYING	ANOTHER	IS	WITH
11	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
HAVE	THE	OF	IS	I	THAT	TWO	WAY	VERY	SO
BE	THIS	IN	ARE	YOU	WHAT	ONE	TIME	MORE	<UH>
WANT	A	FOR	WAS	WE	HOW	THREE	THING	LITTLE	IF
HAD	THAT	ON	HAS	THEY	WHERE	HUNDRED	LOT	MUCH	<UM>
GET	THESE	WITH	WERE	LET	WHEN	M	QUESTION	GOOD	<PARTIAL>
LIKE	MY	AT	GOES	LET'S	IF	T	KIND	DIFFERENT	NOW
GOT	OUR	FROM	HAD	HE	WHY	FIVE	POINT	THAN	THEN
NEED	YOUR	BY	COMES	I'LL	WHICH	D	CASE	IMPORTANT	OKAY
TRY	THOSE	ABOUT	MEANS	PEOPLE	AS	YEARS	IDEA	LONG	WELL
TAKE	THEIR	AS	SAYS	I'D	BECAUSE	FOUR	PROBLEM	AS	BUT

Table 3.2: The top 10 words from the 19 syntactic states from the Lectures* dataset.

3.5 Experiments

To evaluate the effectiveness of models derived from the separation of syntax from content, we performed experiments that compare the perplexities and WERs of various model combinations. For a baseline, we used an adapted model (L+T) that linearly interpolates trigram models trained on the Lectures* (L) and Textbook (T) datasets. In all models, all interpolation weights and additional parameters are tuned on CS-Dev and evaluated with CS-Test (see Section 2.2.1). Unless otherwise noted, modified Kneser-Ney discounting with fixed parameters [22] is applied with the respective training set vocabulary using the SRILM toolkit [138]. Recognition experiments are performed using the SUMMIT speech recognizer (see Section 2.2.2) with an earlier version of the speaker-independent acoustic model trained using minimum classification error discriminative training [108].

3.5.1 Lecture Style

In general, an n-gram model trained on a limited set of topic-specific documents tends to over-emphasize words from the observed topics instead of evenly distributing weights over all potential topics. Specifically, given the list of words following an n-gram context, we would like to de-emphasize the occurrences of observed topic words and ideally redistribute these counts to all potential topic words. As an approximation, we can build such a topic de-emphasized style trigram model by using counts of only n-gram sequences that do not end on a topic word, smoothed over the Lectures* vocabulary. Table 3.3 shows the n-grams corresponding to an utterance used to build the style trigram model. Note that the counts of topic to style word transitions are not altered as these probabilities are mostly independent of the observed topic distribution.

<S> FOR THE SPATIAL MEMORY </S>	
Unigrams:	FOR, THE, SPATIAL, MEMORY, </S>
Bigrams:	<S> FOR, FOR THE, ~~THE SPATIAL~~, ~~SPATIAL MEMORY~~, MEMORY </S>
Trigrams:	<S> <S> FOR, <S> FOR THE, ~~FOR THE SPATIAL~~ ~~THE SPATIAL MEMORY~~, SPATIAL MEMORY </S>

Table 3.3: Style model n-grams. Topic words in the utterance are underlined.

<S> HUFFMAN CODE CAN BE REPRESENTED AS A BINARY TREE ...	
Unigrams:	HUFFMAN, CODE, CAN, BE, REPRESENTED, AS, BINARY, TREE, ...
Bigrams:	<S> HUFFMAN, HUFFMAN CODE (2×), CODE CAN (2×), CAN BE, BE REPRESENTED, REPRESENTED AS, A BINARY, BINARY TREE (2×), ...
Trigrams:	<S> <S> HUFFMAN, <S> HUFFMAN CODE (2×), HUFFMAN CODE CAN (2×), CODE CAN BE (2×), CAN BE REPRESENTED, BE REPRESENTED AS, REPRESENTED AS A, AS A BINARY, A BINARY TREE (2×), ...

Table 3.4: Domain model n-grams. Topic words in the utterance are underlined.

By interpolating the style model (S) from above with the smoothed trigram model based on the Lectures* dataset (L), the combined model (L+S) achieves a 3.6% perplexity reduction and 0.5% absolute WER reduction over the lecture model (L) on the test set, as shown in Table 3.5. Without introducing any topic-specific training data, we can already improve the generic lecture LM performance using the HMM-LDA labels.

3.5.2 Topic Domain

Unlike Lectures*, the Textbook dataset contains content words relevant to the target lectures, but in a mismatched style. Typically, the Textbook trigram model is interpolated with the generic background model to improve the probability estimates of the transitions involving topic words. The interpolation weight is chosen to best fit the probabilities of these n-gram sequences while minimizing the mismatch in style. However, with only one parameter, all n-gram contexts must share the same mixture weight. Because transitions from contexts containing topic words are rarely observed in the off-topic Lectures*, the Textbook model should ideally have higher weight in these contexts than contexts that are more equally observed in both datasets.

One heuristic approach for adjusting the weight in these contexts is to build a topic-domain trigram model (D) from the Textbook n-gram counts with Witten-Bell smoothing [152], where we emphasize the sequences containing a topic word in the n-gram context by doubling their counts. In effect, this reduces the smoothing on words following topic contexts with respect to lower-order models without significantly affecting the transitions from non-topic words. Table 3.4 shows the adjusted counts for an utterance used to build the domain trigram model.

Empirically, interpolating the lectures, textbook, and style models with the domain model (L+T+S+D) further decreases the perplexity by 1.4% and WER by 0.1% over (L+T+S), validating our intuition. Overall, the addition of the style and domain models reduces perplexity and WER by a noticeable 7.1% and 1.0%, respectively, as shown in Table 3.5.

Model	Perplexity Development		Perplexity Test		Word Error Rate Development	Word Error Rate Test
L: Lectures Trigram	180.2	(0.0%)	199.6	(0.0%)	49.5%	50.2%
T: Textbook Trigram	291.7	(+61.8%)	331.7	(+66.2%)		
S: Style Trigram	207.0	(+14.9%)	224.6	(+12.5%)		
D: Domain Trigram	354.1	(+96.5%)	411.6	(+106.3%)		
L+S	174.2	(−3.3%)	192.4	(−3.6%)	49.2%	49.7%
L+T: Baseline	138.3	(0.0%)	154.4	(0.0%)	46.6%	46.7%
L+T+S	131.0	(−5.3%)	145.6	(−5.7%)	46.0%	45.8%
L+T+S+D	128.8	(−6.9%)	143.6	(−7.1%)	45.8%	45.7%
L+T+S+D+Topic100						
• Static Mixture (oracle)	118.1	(−14.6%)	131.3	(−15.0%)	45.5%	**45.4%**
• Dynamic Mixture	**115.7**	(−16.4%)	**129.5**	(−16.1%)	**45.4%**	45.6%

Table 3.5: Perplexity and WER performance of various model combinations. Relative reductions are shown in parentheses.

Huffman tree	Monte Carlo	time segment	assoc key
relative frequency	rand update	the agenda	the table
relative frequencies	random numbers	segment time	local table
the tree	trials remaining	current time	a table
one hundred	trials passed	first agenda	of records

Table 3.6: Sample of n-grams from select topics.

3.5.3 Textbook Topics

In addition to identifying content words, HMM-LDA also assigns words to a topic based on their distribution across documents. Thus, we can apply HMM-LDA with 100 topics to the Textbook dataset to identify representative words and their associated contexts for each topic. From these labels, we can build unsmoothed trigram language models (Topic100) for each topic from the counts of observed n-gram sequences that end in a word assigned to the respective topic.

Table 3.6 shows a sample of the word n-grams identified via this approach for a few topics. Note that some of the n-grams are key phrases for the topic while others contain a mixture of syntactic and topic words. Unlike bag-of-words models that only identify the unigram distribution for each topic, the use of context-dependent labels enables the construction of n-gram topic models that not only characterize the frequencies of topic words, but also describe the transition contexts leading up to these words.

3.5.4 Topic Mixtures

Since each target lecture generally only covers a subset of the available topics, it will be ideal to identify the specific topics corresponding to a target lecture and assign those topic models more weight in a linearly interpolated mixture model. As an ideal case, we performed an oracle experiment to measure the best performance of a statically interpolated topic mixture model (L+T+S+D+Topic100) where we tuned the mixture weights of all mixture components, including the lectures, textbook, style, domain, and the 100 individual topic trigram models on individual target lectures.

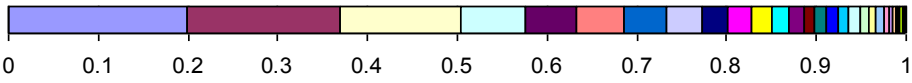


Figure 3-3: Topic mixture weight breakdown.

Table 3.5 shows that by weighting the component models appropriately, we can reduce the perplexity and WER by an additional 7.9% and 0.3%, respectively, over the (L+T+S+D) model even with simple linear interpolation for model combination.

To gain further insight into the topic mixture model, we examine the breakdown of the normalized topic weights for a specific lecture. As shown in Figure 3-3, of the 100 topic models, 15 of them account for over 90% of the total weight. Thus, lectures tend to show a significant topic skew which topic adaptation approaches can model effectively.

3.5.5 Topic Adaptation

Unfortunately, since different lectures cover different topics, we generally cannot tune the topic mixture weights ahead of time. One approach, without any a priori knowledge of the target lecture, is to adaptively estimate the optimal mixture weights as we process the lecture [49]. However, since the topic distribution shifts over a long lecture, modeling a lecture as an interpolation of components with fixed weights may not be the most optimal. Instead, we employ an exponential decay strategy where we update the current mixture distribution by linearly interpolating it with the posterior topic distribution given the current word. Specifically, applying Bayes' rule, the probability of topic t generating the current word w is given by:

$$p(t|w) = \frac{p(w|t)p(t)}{\sum_{t'} p(w|t')p(t')}$$

To achieve the exponential decay, we update the topic distribution after each word according to $p_{i+1}(t) = (1-\alpha)p_i(t) + \alpha p(t|w_i)$, where α is the adaptation rate.

We evaluated this approach of dynamic mixture weight adaptation on the (L+T+S+D+Topic100) model, with the same set of components as the oracle experiment with static weights. As shown in Table 3.5, the dynamic model actually outperforms the static model by more than 1% in perplexity, by better modeling the dynamic topic substructure within the lecture.

To run the recognizer with a dynamic LM, we rescored the top 100 hypotheses generated with the (L+T+S+D) model using the dynamic LM. The WER obtained through such n-best rescoring yielded noticeable improvements over the (L+T+S+D) model without a priori knowledge of the topic distribution, but did not beat the optimal static model on the test set.

To further gain an intuition for mixture weight adaptation, we plot the normalized adapted weights of the topic models across the first lecture of the test set in Figure 3-4. Note that the topic mixture varies greatly across the lecture. In this particular lecture, the lecturer starts out with a review of the previous lecture. Subsequently, he shows an example of computation using accumulators. Finally, he focuses the lecture on the stream as a data structure, with an intervening example that finds pairs of i and j that sum up to a prime. By comparing the topic labels in Figure 3-4 with the top words from the corresponding topics in Table 3.7, we observe that the topic weights obtained via dynamic adaptation match the subject matter of the lecture fairly closely.

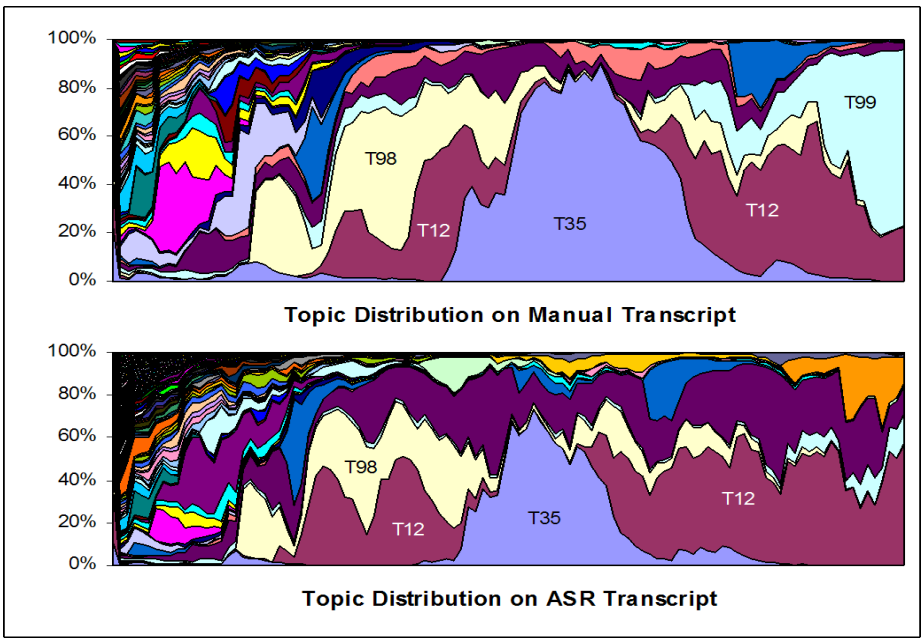


Figure 3-4: Distribution of the topic model weights over the duration of a lecture using manual and ASR transcription.

T12	T35	T98	T99
STREAM	PAIRS	SEQUENCE	OF
S	I	ENUMERATE	SEE
STREAMS	J	ACCUMULATE	AND
INTEGERS	K	MAP	IN
SERIES	PAIR	INTERVAL	FOR
PRIME	S	FILTER	VS
FILTER	INTEGERS	SEQUENCES	REGISTER
DELAYED	SUM	OPERATIONS	DATA
INTERLEAVE	QUEENS	ODD	AS
INFINITE	T	NIL	MAKE

Table 3.7: Top 10 words from select Textbook topics appearing in Figure 3-4.

Finally, to assess the effect that recognition errors have on adaptation performance, we applied the adaptation algorithm to the corresponding transcript from the automatic speech recognizer (ASR). Traditional cache language models tend to be vulnerable to recognition errors since incorrect words in the history negatively bias the prediction of the current word. However, by adapting at a topic level, which reduces the number of dynamic parameters, the dynamic topic model is less sensitive to recognition errors. As seen in Figure 3-4, even with a word error rate of around 40%, the normalized topic mixture weights from the ASR transcript still show a strong resemblance to the original weights from the manual reference transcript.

3.6 Summary

In this chapter, we have shown how to leverage context-dependent state and topic labels, such as the ones generated by the HMM-LDA model, to construct better language models for lecture transcription and extend topic models beyond traditional unigrams. Although the WER of the best model still exceeds 45%, by dynamically updating the mixture weights to model the topic substructure within individual lectures, we are able to reduce the test set perplexity and WER by over 16% and 1.1%, respectively, relative to the combined Lectures* and Textbook (L+T) baseline.

The experiments in this chapter combine models through simple linear interpolation. As motivated in Section 3.5.2, allowing for context-dependent interpolation weights based on topic labels may yield further improvements in both perplexity and WER. Likewise, heuristically de-emphasizing the probabilities of n-grams ending on topic words can yield significant performance improvements without introducing additional data. Thus, in the next two chapters, we will extend these intuitions and examine more principled approaches to learning context-dependent interpolation weights and de-emphasizing the weights of out-of-domain n-grams.

Chapter 4

Generalized Linear Interpolation

4.1 Motivation

With the increasing focus of speech recognition and natural language processing applications on domains with limited amount of in-domain training data, enhanced system performance often relies on approaches involving model combinations. For language modeling, we can improve the estimation of the underlying n-gram probabilities from limited data using modeling techniques such as class n-grams [13,80,105,161] and topic mixtures [25,34,49,79]. We can also gather additional sources of related data from the web [14, 15, 95, 110, 131, 160] and even utilize the initial recognition hypotheses [1, 2, 20, 65, 115, 135, 144] for language model (LM) adaptation. (See Section 2.3 for a detailed survey of related work.)

Despite the prevalent use of model combination techniques to improve speech recognition performance on domains with limited data, little prior research has focused on the choice of the actual interpolation method. For merging language models, the most popular approach has been the simple linear interpolation [81], largely due to its simplicity and toolkit support. Although count merging [1] has been demonstrated to generally yield better performance, the interpolation weights should really depend on the component LM relevance in addition to the observation counts, especially when combining models trained from disparate sources.

In this work, we propose a generalization of linear interpolation that computes context-dependent mixture weights from a set of n-gram context features. By using multinomial logistic regression to model the interpolation weights, the resulting generalized linear interpolation (GLI) model subsumes both linear interpolation (LI) and count merging (CM) as special cases. On the lecture transcription task, GLI achieves up to 0.8% and 0.3% word error rate (WER) improvement over LI and CM using n-gram counts and other features that correlate with the topic specificity of the n-gram context. As tuning the model parameters with as little as 500 words of in-domain development set outperforms equivalently trained LI and CM models, generalized linear interpolation provides a practical approach to utilizing various n-gram context features towards improving n-gram language model interpolation.

We start this chapter by first described existing interpolation methods. Next, we present the generalized linear interpolation technique, its relationship with linear interpolation and count merging, and the n-gram context features considered in this work. After a comparative evaluation of the performance, we perform a detailed analysis of the tuned parameters and the model performance as a function of various experimental conditions. We conclude with a summary of the results and present various ideas for future work.

4.2 Related Work

Given a set of training texts, we can build a combined language model using multiple techniques. One of the simplest techniques, sometimes referred to as the brute-force approach, is to merge all texts and train a single smoothed n-gram LM. Because the training corpora often differ in size and relevance to the target domain, simply summing the n-gram counts rarely achieves the best result.

Instead, linear interpolation (see Section 2.3.2), the most popular model combination technique, first trains individual n-gram LMs from each corpus. Given the resulting set of n-gram language models, it computes the weighted average of the component model probabilities $p^{\text{LI}}_{\text{dynamic}}(w|h) = \sum_i \lambda_i p_i(w|h)$, where $p_i(w|h)$ is the smoothed probability of word w following n-gram history h according to component model i. The interpolation weight λ_i, satisfying $\lambda_i \geq 0$ and $\sum_i \lambda_i = 1$, is typically tuned to optimize the development set perplexity. Instead of dynamically interpolating the component probabilities, we statically interpolate the models in this work to yield a single n-gram backoff model, which generally obtains better performance. Specifically,

$$p^{\text{LI}}(w|h) = \begin{cases} \sum_i \lambda_i p_i(w|h) & \text{if } \sum_i c_i(hw) > 0 \\ \alpha(h) p^{\text{LI}}(w|h') & \text{otherwise} \end{cases}$$

where $c_i(hw)$ is the observation count of the n-gram hw for model i, $\alpha(h)$ is the history-dependent backoff weight chosen such that $\sum_w p^{\text{LI}}(w|h) = 1$, and h' is the truncated backoff history for history h.

An n-gram language model consists of not a single model, but rather a collection of word distributions, one for each n-gram history. As each conditional probability $p(w|h)$ is estimated from different number of n-grams with history h, estimates derived from higher counts are more robust and reliable than ones computed from fewer observations. Thus, instead of interpolating all conditional probabilities with fixed weights, the interpolation weights should be a function of the n-gram history that considers the reliability of the component estimates via the observation counts and the relevance of each component LM under the specific n-gram context.

Instead of taking the weighted average of the component n-gram probabilities, count merging, also known as count mixing and MAP adaptation, computes the overall probability from a weighted sum of the discounted component n-gram counts. Extending the two-component formulation presented in previous works [1, 38], count merging with n-gram backoff computes the following:

$$p^{\text{CM}}_{\text{dynamic}}(w|h) = \frac{\sum_i \beta_i c_i^{\text{disc}}(hw)}{\sum_j \beta_j \sum_w c_j(hw)}$$

$$p^{\text{CM}}(w|h) = \begin{cases} p^{\text{CM}}_{\text{dynamic}}(w|h) & \text{if } \sum_i c_i(hw) > 0 \\ \alpha(h) p^{\text{CM}}(w|h') & \text{otherwise} \end{cases}$$

where $c_i(hw)$ is the observation count of the n-gram hw in the i^{th} component, $c_i^{\text{disc}}(hw)$ is the corresponding discounted count after smoothing, β_i is the component model weighting factor, and $\alpha(h)$ is the backoff weight. As scaling all weighting factors β_i by a non-zero constant yields an identical model, we will set $\beta_1 = 1$ by convention to constrain the parameter space. Similar to linear interpolation, the model parameters β_i can be tuned to minimize the perplexity of a disjoint development set.

A smoothed n-gram LM assigns probability $p(w|h) = c^{\text{disc}}(hw)/\sum_w c(hw)$ to observed n-grams.[1] Thus, by substituting and refactoring the terms:

$$p^{\text{CM}}(w|h) = \frac{\sum_i \beta_i c_i^{\text{disc}}(hw)}{\sum_j \beta_j \sum_w c_j(hw)} = \frac{\sum_i \beta_i \left[\sum_w c_i(hw)\right] p_i(w|h)}{\sum_j \beta_j \sum_w c_j(hw)}$$
$$= \sum_i \left[\frac{\beta_i \sum_w c_i(hw)}{\sum_j \beta_j \sum_w c_j(hw)}\right] p_i(w|h)$$

we see that count merging is simply a generalization of linear interpolation, where the interpolation weight $\lambda_i(h) = \beta_i \sum_w c_i(hw)/\sum_j \beta_j \sum_w c_j(hw)$ now depends on the n-gram history via its observed count. Thus, instead of a constant interpolation weight, count merging applies an interpolation weight proportional to the total number of observances of n-grams with history h. The more data used to train the word distribution following h, the more we trust and weight the resulting estimate.[2]

Choosing interpolation weights proportional to the amount of observed data addresses part of the deficiencies with linear interpolation. However, since count merging scales the interpolation weights according to the observation count ratio, it will use the same set of weights for an n-gram history that is observed 10 and 1000 times as another history that is observed 10 billion and 1 trillion times. Intuitively, since the component estimates for the latter history are already reliably trained, its weights should depend primarily on the overall component relevance and not the observation count ratio. This suggests that the optimal interpolation weights depend on the observation counts in a non-linear fashion.

Instead of using fixed weights to interpolate probabilities or counts, past work has also examined computing context-dependent interpolation weights as a function of the n-gram history [21, 157]. For example, we can cluster the n-gram histories into equivalence classes and tune a set of independent interpolation weights for each class using the Expectation-Maximization (EM) algorithm [33,130]. However, as the number of free parameters increases with the number of classes, this approach generally requires a large of amount of in-domain development data and is impractical in our target scenario with limited matched data.

Previous research has also explored other techniques to obtain context-dependent interpolation weights without introducing a large number of free parameters. For example, Zhou et al. [157] proposed a heuristic motivated by Witten-Bell smoothing [152] that considers the number of unique words following an n-gram when computing the interpolation weight. As an alternative, Chen and Huang [21] presented a *fuzzy controller* that adjusts the interpolation weights based on heuristic measures of estimate confidence, word importance, and probability ratio. Unfortunately, both techniques assume the existence of an in-domain model and avoid parameter tuning. Intuitively, we should be able to achieve better performance by utilizing a more principled model to combine the n-gram history features and tune the model parameters on a matched development set.

[1] Technically, this is only true for non-interpolated smoothing techniques that only backoff to the lower-order model for unseen n-grams. In practice, the difference is generally insignificant after parameter tuning.

[2] Although it is tempting to assume that $c(h) = \sum_w c(hw)$, this is not completely accurate if we modify the observation counts as with Kneser-Ney smoothing (see Section 2.1.3). Unfortunately, we made this mistake when evaluating the performance of count merging in our published papers [70, 73–75]. However, the performance difference between the two variations is generally small and not statistically significant. Furthermore, we can interpret this flawed implementation of count merging as just another special case of generalized linear interpolation.

4.3 Generalized Linear Interpolation

4.3.1 Model

In this work, we propose the generalized linear interpolation (GLI) technique that models the interpolation weights as a function of n-gram history features predictive of the component relevance instead of using fixed constants across all n-gram contexts. Specifically, given component n-gram models $p_i(w|h)$, generalized linear interpolation computes:

$$p^{\text{GLI}}_{\text{dynamic}}(w|h) = \textstyle\sum_i \lambda_i(h) p_i(w|h)$$

$$p^{\text{GLI}}(w|h) = \begin{cases} p^{\text{GLI}}_{\text{dynamic}}(w|h) & \text{if } \sum_i c_i(hw) > 0 \\ \alpha(h) p^{\text{GLI}}(w|h') & \text{otherwise} \end{cases}$$

where $\lambda_i(h) \geq 0$ are the context-dependent interpolation weights satisfying $\sum_i \lambda_i(h) = 1$, $c_i(hw)$ is the observation count of n-gram hw in component i, and $\alpha(h)$ is computed such that $\sum_w p^{\text{GLI}}(w|h) = 1$.

In this work, we will model the interpolation weight function using the output from a multinomial logistic regression model [7, 120]. Specifically, given a set of component-dependent n-gram history features $\boldsymbol{\Psi}_i(h) = \left[\Psi_i^1(h) \cdots \Psi_i^{F_i}(h)\right]$ computed for each n-gram history h in the interpolated model, we model the context-dependent interpolation weights as:

$$\lambda_i(h) = \frac{\exp[\boldsymbol{\Psi}_i(h) \cdot \boldsymbol{\theta}_i]}{Z(h)} \qquad Z(h) = \sum_i \exp[\boldsymbol{\Psi}_i(h) \cdot \boldsymbol{\theta}_i]$$

where $\boldsymbol{\theta}_i$ is the parameter vector for component i and $Z(h)$ is the normalization factor chosen to ensure that $\sum_i \lambda_i(h) = 1$. This is also commonly known as the softmax function.

Although we can model the interpolation weight function using other parametric and even non-parametric forms, multinomial logistic regression exhibits the following desirable properties. First, the resulting interpolation weights $\lambda_i(h)$ are guaranteed to satisfy the non-negativity and normalization constraints, regardless of the choice of the parameter vectors $\boldsymbol{\theta}_i$. Since the development set perplexity has an easy-to-compute closed-form expression for the gradient, we can efficiently apply gradient-based numerical optimization techniques to tune the model parameters.

For clarity, we can rewrite the interpolation weight expression as:

$$\lambda_i(h) = \frac{\exp[\boldsymbol{\Psi}_i(h) \cdot \boldsymbol{\theta}_i + \gamma_i]}{Z(h)}$$

where we have explicitly factored out a bias parameter γ_i, equivalent to the parameter θ_i^0 associated with a constant feature $\Psi_i^0(h) = 1$. In this model, we can interpret the bias parameter γ_i as the overall relevance of component i. The feature parameter θ_i^f corresponds to the effect feature f has on the interpolation weight of component i. A positive value of θ_i^f increases the relative weight of component i for n-gram histories with positive feature values $\Psi_i^f(h)$. Since a constant offset to all γ_i's yields identical interpolation weights due to cancellation, we will set $\gamma_1 = 0$ to constrain the solution space. Although we can tune independent parameters $\boldsymbol{\theta}_i$ and γ_i for each n-gram order of the language model, in general, we tie the model parameters across order to improve the robustness of the estimated parameters in scenarios with limited development set data.

4.3.2 Relationship with LI and CM

As defined, generalized linear interpolation subsumes both linear interpolation and count merging as special cases. Specifically, by setting the feature vectors to an empty vector $\boldsymbol{\Psi}_i(h) = [\,]$, generalized linear interpolation degenerates to the simple linear interpolation, where $\lambda_i \propto e^{\gamma_i}$.

$$\lambda_i(h) = \frac{\exp[\boldsymbol{\Psi}_i(h) \cdot \boldsymbol{\theta}_i + \gamma_i]}{Z(h)} = \frac{\exp[\gamma_i]}{Z(h)} \propto e^{\gamma_i}$$

Likewise, if we define $\boldsymbol{\Psi}_i(h) = [\log(\sum_w c_i(hw))]$ and $\boldsymbol{\theta}_i = [1]$:

$$\begin{aligned}\lambda_i(h) &= \frac{\exp[\boldsymbol{\Psi}_i(h) \cdot \boldsymbol{\theta}_i + \gamma_i]}{Z(h)} = \frac{\exp[\log(\sum_w c_i(hw)) + \gamma_i]}{Z(h)} \\ &= \frac{e^{\gamma_i}\sum_w c_i(hw)}{Z(h)} = \frac{\beta_i \sum_w c_i(hw)}{Z(h)} = \frac{\beta_i \sum_w c_i(hw)}{\sum_j \beta_j \sum_w c_j(hw)}\end{aligned}$$

we obtain count merging with $\beta_i = e^{\gamma_i}$. To achieve exact equivalence, we allow zero counts in the computation by defining $\exp(\log 0 + \cdots)/Z(h) = \exp(-\infty) = 0$.[3]

By choosing the parametric family of the interpolation weight function to include linear interpolation and count merging as special cases, we guarantee that the resulting model will at least match if not almost always exceed the performance of the existing techniques on the development set. Given a sufficiently large development set to avoid overfitting, this gain generally will also carry over to unseen test sets.

4.3.3 Features

Generalized linear interpolation enables arbitrary n-gram history features to be utilized in the computation of the interpolation weight. Thus, we can apply feature engineering techniques to design, select, and combine features. In this chapter, we will consider two classes of features. The first set of features are ones that can be derived automatically from n-gram counts, whereas the second set utilizes the document segmentation information often provided with the text corpora to predict component relevance.

Count Features

Let us define the left-branching count $c^l(h) = N_{1+}(\bullet h) = |\{w : c(wh) > 0\}|$ as the number of unique words that appear before h, following the notation in [22]. Recall from Section 2.1.3 that this is the count used in Kneser-Ney smoothing [90] to estimate the probability of lower-order models. Symmetrically, define the right-branching count $c^r(h) = N_{1+}(h\bullet) = |\{w : c(hw) > 0\}|$ as the number of unique words that appear after h, as used in Witten-Bell smoothing [152].

As shown in the previous section, we use the log-count feature $\boldsymbol{\Psi}_i(h) = [\log(\sum_w c_i(hw))]$ to derive count merging from generalized linear interpolation. For Kneser-Ney and related smoothing techniques, the counts used in the lower-order models are replaced by the left-branching counts. Thus, for clarify, we will notate the effective counts used for probability estimation as $\tilde{c}_i(hw)$. Note that in general, $\tilde{c}_i(h) \neq \sum_w \tilde{c}_i(hw)$. Thus, using $\boldsymbol{\Psi}_i(h) = [\log \tilde{c}_i(h))]$ as features is not strictly equivalent to count merging when using Kneser-Ney smoothing.

[3] In practice, we can approximate this by adding a small constant when taking the log, i.e., $\log(x + 10^{-99})$.

In this chapter, we consider the following count-based features to estimate the robustness and relevance of an n-gram component for history h:

$$\log(\textstyle\sum_w c(hw)) \qquad \log(c^l(h)) \qquad \log(c^r(h)) \qquad \log(\textstyle\sum_w \tilde{c}(hw))$$

In addition to the above log counts, we also include corresponding features using $\log_{+1}(x) \equiv \log(x+1)$ to obtain non-negative values and polynomial kernels for better fit. Specifically, we will examine the log-squared features computed using $\log^2_{+1}(x) \equiv (\log(x+1))^2$, as it increases monotonically for $x \geq 0$.

Topic Features

As motivated and empirically observed in Section 3.5.2, choosing interpolation weights that depend on the topic specificity of the n-gram history improves the model performance. There, we heuristically doubled the count of an n-gram from the Textbook corpus whose history contains a word labeled as a HMM-LDA topic word [67] in order to increase the effective weight of the Textbook component in that context. In this chapter, we apply similar intuitions using the more mathematically principled generalized linear interpolation with HMM-LDA and other n-gram features indicative of the n-gram history topic specificity.

As discussed in more detail in Chapter 5, a training corpus is often segmented into documents. To estimate the topic specificity of an n-gram, we can measure the concentration of the n-gram across documents. To guide the presentation of the n-gram features below, we will consider the following example partition of the training corpus. Words tagged by HMM-LDA as topic words appear in bold.

A B A	A **C C** A	**B** A B
B A **A** C	C B A	A B A
A C B	A A **C**	A **B B** A

One way to estimate the specificity of an n-gram history h across partitions is to measure the n-gram frequency $F(h)$, or the fraction of partitions containing h. For instance, $F(\text{A}) = 3/3$, $F(\text{C}) = 2/3$. However, as the size of each partition increases, this ratio increases to 1, since most n-grams have a non-zero probability of appearing in each partition. Thus, an alternative is to compute the normalized entropy of the n-gram history distribution across the S partitions, or $H(h) = \frac{-1}{\log S} \sum_{s=1}^{S} p_s(h) \log p_s(h)$, where $p_s(h)$ is the fraction of the history h that is observed in partition s. Unseen histories are assigned the value 0. For example, the normalized entropy of the unigram C is $H(\text{C}) = \frac{-1}{\log 3}[\frac{2}{6}\log\frac{2}{6} + \frac{4}{6}\log\frac{4}{6} + 0] = 0.58$. n-gram histories clustered in fewer partitions have lower entropy than ones that are more evenly spread out.

Following Section 3.5.2, we also consider features derived from the HMM-LDA word topic labels.[4] Specifically, we define the n-gram history topic probability $T^h(h)$ as the empirical probability that any word in the history h is labeled as a topic word. Similarly, we define the n-gram word topic probability $T^w(h)$ as the empirical probability that the n-gram history h is followed by a topic word. The probabilities for unobserved histories are assigned the value 0. Intuitively, the frequency with which a topic word follows an

[4]HMM-LDA is performed using 20 states and 50 topics with a 3rd-order HMM. Hyperparameters are sampled with a log-normal Metropolis proposal. The model with the highest likelihood from among 10,000 iterations of Gibbs sampling is used.

Feature $\Psi(h)$	$\emptyset$	<s>	a	vector	abstract	of the	i think	the sun	k means
$\log(\sum_w c(hw))$	14.57	11.77	10.80	7.09	4.01	9.29	8.09	5.86	3.47
$\log(c^l(h))$	10.30	9.33	7.92	4.82	3.33	7.55	6.37	4.38	2.83
$\log(c^r(h))$	10.30	8.60	8.57	5.15	3.64	7.97	5.95	4.49	2.77
$\log(\sum_w \tilde{c}(hw))$	12.98	11.77	10.18	6.40	3.97	9.29	8.09	5.86	3.47
$\log_{+1}(\sum_w \tilde{c}(hw))$	12.98	11.77	10.18	6.40	3.99	9.29	8.09	5.86	3.50
$\log^2_{+1}(\sum_w \tilde{c}(hw))$	168.46	138.46	103.61	40.96	15.92	86.30	65.38	34.38	12.23
$F(h)$	0.00	1.00	1.00	0.36	0.12	1.00	0.93	0.18	0.00
$H(h)$	0.00	0.97	0.98	0.70	0.57	0.96	0.84	0.56	0.00
$T^h(h)$	0.00	0.00	0.03	1.00	0.11	0.00	0.00	1.00	1.00
$T^w(h)$	0.19	0.09	0.39	0.30	0.25	0.70	0.05	0.06	0.41

Table 4.1: A list of generalized linear interpolation features. c: count, c^l: left-branching count, c^r: right-branching count, $\tilde{c}$: effective count; F: n-gram frequency, H: normalized entropy, T^h: n-gram history topic probability, T^w: n-gram word topic probability.

n-gram history within a corpus is predictive of the relevance of that component. On-topic components should receive higher weight in contexts often preceding in-domain topic words. Conversely, off-topic components should receive lower weight when they are likely to predict an out-of-domain topic word.

In the example corpus, $T^h(\emptyset) = 0$, $T^h(\text{C}) = 3/6$, $T^h(\text{A C}) = 3/4$. $T^w(\emptyset) = 8/39$, $T^w(A) = 4/15$, $T^w(\text{A C}) = 1/4$, $T^w(\text{B C}) = 0/0 = 0$. Table 4.1 lists all the n-gram count and topic features examined in this chapter and their values on a select subset of n-gram histories from the Lectures corpus.

4.3.4 Training

Traditionally, the linear interpolation weights are tuned to minimize the development set perplexity using the EM algorithm. With count merging, the scaling parameters are selected empirically by trial and error techniques, with some previous work suggesting that it cannot be easily optimized [1, 48]. In Chapter 7, we present an efficient n-gram data structure and optimized algorithms specifically designed for iterative language model estimation. With such an implementation, we can tune the feature and bias parameters of generalized linear interpolation using any numerical optimization technique [123] to iteratively estimate the optimal parameter values. As generalized linear interpolation encompasses linear interpolation and count merging, these models can also be tuned using the same numerical optimization framework.

For all experiments in this chapter, the bias and feature parameters are initialized to 0 and 1, respectively, and tuned to minimize the development set perplexity using the limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) unconstrained optimization technique [123,159], a quasi-Newton method that uses the second derivative Hessian matrix iteratively estimated from the gradients evaluated along the search path to improve the convergence towards the function minimum. For linear interpolation, we were actually able to achieve convergence in fewer iterations and less time using L-BFGS than the traditional EM algorithm.

Model	Perplexity		Word Error Rate	
	Development	Test	Development	Test
Textbook	283.4	326.1	41.9%	42.4%
Lectures	168.6	189.3	34.8%	36.8%
Switchboard	301.9	299.1	42.3%	42.5%
Textbook + Lectures				
LI	125.6	141.6	31.7%	33.4%
BF	132.1 (+5.1%)	150.4 (+6.3%)	31.7%	33.4%
CM	122.3 (-2.7%)	138.1 (-2.5%)	31.4%	32.9%
GLI	121.7 (-3.1%)	137.3 (-3.0%)	31.3%	32.9%

Table 4.2: Interpolation model performance. LI: Linear interpolation, BF: Brute-force interpolation, CM: Count merging, GLI: Generalized linear interpolation. Both CM and GLI are configured with $\Psi_i(h) = \log(\sum_w \tilde{c}_i(hw))$.

4.4 Experiments

4.4.1 Setup

In this section, we compare the performance of generalized linear interpolation against a few existing interpolation techniques by evaluating the perplexity and recognition WER on the lecture transcription task (see Section 2.2.1). For each training corpus (Textbook, Lectures, Switchboard), we build a trigram language model using modified Kneser-Ney smoothing [22] with the default corpus-specific vocabulary. As models interpolated over the same components share a common vocabulary regardless of the interpolation technique, we can fairly compare the perplexities computed over n-grams with non-zero probabilities. As the interpolated language models can be encoded directly as an n-gram backoff models, they can be applied directly in the first recognition pass. However, in these experiments, we compute the WER by rescoring the recognition lattices generated using the count merging LM. The WER difference resulting from the use of lattice rescoring is around 0.01%.

4.4.2 Baseline

In Table 4.2, we summarize the performance of various interpolation techniques on the lecture transcription task. With Textbook and Lectures, all model combination techniques achieve significantly lower perplexity (though not strictly comparable due to differences in vocabulary) and reduced WER over the individual component models. Compared with the linear interpolation (LI) baseline with a single interpolation parameter, the brute-force (BF) text concatenation method results in worse perplexity and no change in WER. However, consistent with observations from previous work [1, 39], count merging (CM) outperforms linear interpolation by 0.5% test set WER, with $p < 0.001$ using the Matched Pairs Sentence Segment Word Error significance test [51].

As shown in Section 4.3.2, count merging is just a special case of generalized linear interpolation (GLI) that uses the effective count associated with each n-gram history as feature, or $\boldsymbol{\Psi}_i(h) = [\log(\sum_w c_i(hw))]$. However, unlike GLI, CM constrains the feature parameters to $\boldsymbol{\theta}_i = [1]$. By relaxing the constraint and tuning the parameters to minimize the development set perplexity, GLI achieves a small, although statistically insignificant, reduction in both perplexity and WER on the same feature.

Feature $\mathbf{\Psi}_i(h)$	Perplexity Development	Perplexity Test	Word Error Rate Development	Word Error Rate Test
None (LI)	125.6	141.6	31.7%	33.4%
+ $\log(\sum_w \tilde{c}(hw))$	121.7 (-3.1%)	137.3 (-3.0%)	*31.3%*	*32.9%*
+ $\log(\sum_w c(hw))$	121.8 (-3.1%)	137.3 (-3.0%)	*31.3%*	*32.8%*
+ $\log(c^l(h))$	122.1 (-2.8%)	137.8 (-2.7%)	*31.3%*	*32.9%*
+ $\log(c^r(h))$	122.6 (-2.4%)	138.2 (-2.4%)	*31.3%*	*32.9%*
+ $\log_{+1}(\sum_w \tilde{c}(hw))$	120.6 (-4.0%)	135.9 (-4.0%)	*31.3%*	***32.8%***
+ $\log^2_{+1}(\sum_w \tilde{c}(hw))$	**120.6** (-4.0%)	**135.6** (-4.2%)	***31.3%***	*32.9%*
+ $F(h)$	122.4 (-2.5%)	137.7 (-2.7%)	*31.5%*	*33.1%*
+ $H(h)$	121.3 (-3.5%)	136.5 (-3.6%)	*31.3%*	*32.9%*
+ $T^h(h)$	124.7 (-0.8%)	141.0 (-0.4%)	31.6%	33.4%
+ $T^w(h)$	121.8 (-3.1%)	137.3 (-3.1%)	*31.4%*	*33.2%*

Table 4.3: GLI performance with various features. Best perplexity/WER values are in bold. Statistically significant improvements ($p < 0.05$) appear in italics.

Mathematically, the tuned θ_i parameter simplifies to an exponent on the count feature.

$$\lambda_i(h) = \frac{\exp[\mathbf{\Psi}_i(h) \cdot \boldsymbol{\theta}_i + \gamma_i]}{Z(h)} = \frac{\exp[\log(\sum_w c_i(hw)) \cdot \theta_i + \gamma_i]}{Z(h)}$$
$$= \frac{e^{\gamma_i}(\sum_w c_i(hw))^{\theta_i}}{Z(h)} = \frac{\beta_i(\sum_w c_i(hw))^{\theta_i}}{\sum_j \beta_j(\sum_w c_j(hw))^{\theta_i}}$$

Intuitively, given sufficient n-gram histories to yield good estimates of the word distribution, the count should no longer play a significant role in determining the interpolation weight. Thus, we expect the optimal exponent on the count to be less than the fixed value of 1.0 for CM. Empirically, we find this to be the case, with the optimal value of around 0.7.

4.4.3 Features

Generalized linear interpolation allows us to use arbitrary features of the n-gram history. In Table 4.3, we compare the performance of GLI with various n-gram history features. Specifically, we set each GLI component feature $\Psi_i(h)$ to the values computed from the corresponding component data. As GLI reduces to LI when $\theta_i = 0$, the development set perplexity is guaranteed to outperform LI. With a sufficiently large development set, the performance improvement should carry over to the test set without overfitting. Overall, all count-based features reduce the perplexity and achieve a similar WER improvement. Although intuitively well-motivated, the performance of topic-based features varies widely and is overall worse than count-based features when applied in isolation.

In Table 4.4, we evaluate the performance of GLI when combining various features. Although topic features perform poorly in isolation, they perform better than count features when combined with the CM feature $\log(\sum_w \tilde{c}(hw))$. By greedily selecting the feature with the lowest development set WER, we are able to significantly reduce the test set WER from 32.9% to 32.6% ($p < 0.001$) using CM counts $\log(\sum_w \tilde{c}(hw))$, word topic probability $T^w(h)$, and n-gram distribution frequency $F(h)$. With these three features, GLI achieves a statistically significant 0.8% and 0.3% absolute WER reduction over the baseline LI and CM techniques, respectively.

Feature $\mathbf{\Psi}_i(h)$	Perplexity Development	Perplexity Test	Word Error Rate Development	Word Error Rate Test
$\log(\sum_w \tilde{c}(hw))$	121.7	137.3	31.3%	32.9%
$+ \log(\sum_w c(hw))$	121.5 (-0.1%)	137.1 (-0.1%)	31.3%	*32.8%*
$+ \log(c^l(h))$	121.6 (-0.1%)	137.2 (-0.1%)	31.3%	32.9%
$+ \log(c^r(h))$	120.7 (-0.9%)	135.9 (-1.0%)	31.2%	32.8%
$+ \log_{+1}(\sum_w \tilde{c}(hw))$	120.6 (-0.9%)	136.0 (-0.9%)	31.3%	32.9%
$+ \log^2_{+1}(\sum_w \tilde{c}(hw))$	121.1 (-0.5%)	136.5 (-0.6%)	*31.2%*	***32.7%***
$+ F(h)$	121.7 (-0.1%)	137.3 (-0.0%)	31.3%	32.8%
$+ H(h)$	121.6 (-0.1%)	137.2 (-0.1%)	31.2%	32.8%
$+ T^h(h)$	121.7 (-0.0%)	137.3 (+0.0%)	31.3%	32.8%
$+ T^w(h)$	**120.0** (-1.4%)	**135.4** (-1.4%)	***31.1%***	32.8%
$\log(\sum_w \tilde{c}(hw)) + T^w(h)$	120.0	135.4	31.1%	32.8%
$+ \log(\sum_w c(hw))$	119.6 (-0.3%)	134.9 (-0.4%)	31.1%	*32.7%*
$+ \log(c^l(h))$	120.0 (-0.0%)	135.4 (-0.0%)	31.1%	32.8%
$+ \log(c^r(h))$	119.9 (-0.1%)	135.2 (-0.1%)	31.1%	32.7%
$+ \log_{+1}(\sum_w \tilde{c}(hw))$	118.6 (-1.2%)	133.9 (-1.1%)	31.1%	32.9%
$+ \log^2_{+1}(\sum_w \tilde{c}(hw))$	**118.5** (-1.3%)	**133.7** (-1.3%)	31.1%	32.8%
$+ F(h)$	119.7 (-0.2%)	135.1 (-0.2%)	***31.0%***	***32.6%***
$+ H(h)$	119.7 (-0.3%)	135.1 (-0.2%)	31.0%	*32.6%*
$+ T^h(h)$	119.9 (-0.1%)	135.4 (+0.0%)	31.1%	32.8%

Table 4.4: GLI performance with various feature combinations. Best perplexity/WER values are in bold. Statistically significant improvements ($p < 0.05$) appear in italics.

4.5 Analysis

4.5.1 Feature Parameters

To obtain further insight into how generalized linear interpolation improves the resulting n-gram model, we present below the optimized parameter values for the GLI model between Textbook and Lectures using features $\mathbf{\Psi}(h) = [\log(\sum_w \tilde{c}(hw)), T^w(h)]$.

$$\boldsymbol{\theta}_{\text{Textbook}} = [0.64, 0.71] \quad \boldsymbol{\theta}_{\text{Lectures}} = [0.56, -3.01] \qquad \gamma_{\text{Textbook}} = 0.00 \quad \gamma_{\text{Lectures}} = 0.40$$

As discussed in Section 4.4.2, the parameters associated with the n-gram history counts should be less than 1, as observed. For the word topic probability feature $T^w(h)$, values close to 1 correspond to n-gram histories that are likely to be followed by topic words in the component corpus. Thus, the positive parameter value $\theta^{T^w}_{\text{Textbook}} = 0.71$ indicates an emphasis on the Textbook component when the current n-gram context often precedes a topic word in the Textbook corpus. As the Textbook corpus matches the topic of the target lecture, increasing its interpolation weight in these contexts improves the prediction of topic words.

Conversely, the negative parameter value $\theta^{T^w}_{\text{Lectures}} = -3.01$ shows a de-emphasis on the Lectures component when the current n-gram context frequently precedes a topic word in the Lectures corpus. As the Lectures corpus has mismatched topics, decreasing its interpolation weight in these contexts improves the overall fit of the interpolated LM.

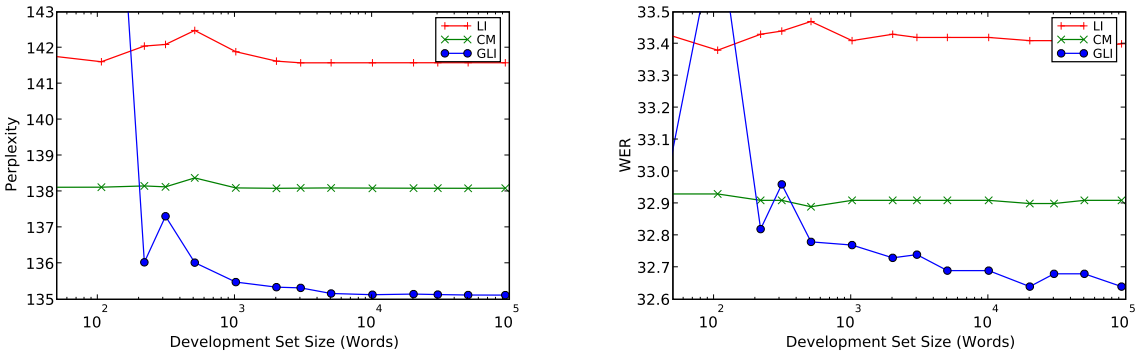


Figure 4-1: Test set performance vs. development set size.

Model	Perplexity / WER T + L	T + S	L + S	T + L + S
LI	141.6 / 33.4%	169.4 / 35.8%	191.6 / 36.5%	139.5 / 33.3%
CM	138.1 / *32.9%*	171.4 / *35.3%*	194.2 / 36.5%	136.9 / *32.9%*
GLI				
+ $\log(\sum_w \tilde{c}(hw))$	137.3 / *32.9%*	168.5 / *35.2%*	191.1 / 36.4%	134.9 / *32.8%*
+ $T^w(h)$	135.4 / ***32.8%***	166.8 / ***35.1%***	191.0 / 36.5%	132.8 / ***32.6%***
+ $F(h)$	135.1 / ***32.6%***	166.7 / ***35.0%***	190.8 / 36.5%	132.5 / ***32.6%***

Table 4.5: Test set performance with various LM combinations. T: Textbook, L: Lectures, S: Switchboard. Statistically significant improvements ($p \leq 0.05$) over LI and CM appear in italics and bold, respectively.

4.5.2 Development Set Size

To obtain a better sense of how GLI performs under different training conditions, we measure the sensitivity of the interpolated model perplexity and WER to the size of the development set. In Figure 4-1, we plot the perplexity and WER of interpolating Textbook and Lectures using LI, CM, and GLI with $\mathbf{\Psi}(h) = [\log(\sum_w \tilde{c}(hw)), T^w(h), F(h)]$. We vary the development set size by randomly selecting utterances from the full development set. As shown, with only 200 words, the test set perplexity of the 7-parameter GLI model already outperforms both LI and CM and converges to within 1 point of the final value after 500 words. Similarly, the WER consistently improves over LI and CM with only 500 words and converges to within 0.1% of the best value beyond 2000 words. These results suggest that the performance gains from the generalized linear interpolation model can be obtained with relatively little in-domain data.

4.5.3 Training Components

Interpolating different combinations of training models generally yields similar trends (Table 4.5), with generalized linear interpolation using a combination of count and topic features significantly outperforming count merging and standard linear interpolation by up to 0.8% absolute reduction in WER. When combining Lectures and Switchboard, however, any improvement over linear interpolation is generally minimal. Given that linear interpolation only reduces WER by 0.3% over the Lectures model, Switchboard's mismatch in both topic and style appears to leave little room for additional improvement over linear interpolation.

Model	# Params	Tied / Independent Perplexity	Word Error Rate
LI	1 / 3	141.6 / 140.5	33.4% / 33.2%
CM	1 / 3	138.1 / 137.7	32.9% / 32.8%
GLI			
+ $\log(\sum_w \tilde{c}(hw))$	3 / 9	137.3 / 137.0	32.9% / 32.8%
+ $T^w(h)$	5 / 15	135.4 / 134.9	32.8% / 32.7%
+ $F(h)$	7 / 21	135.1 / 134.8	32.6% / 32.7%

Table 4.6: Test set performance with independent parameters across n-gram order.

4.5.4 Parameter Tying

In all the experiments presented so far, we tied all parameter values across n-gram orders to reduce the number of tunable parameters and improve the robustness of the value estimates. In scenarios where sufficient matched development data is available, we can consider using independent bias and feature parameters across n-gram orders.

In Table 4.6, we compare the test set performance of various interpolated models with tied and independent parameters. By allowing distinct parameters across n-gram orders, we are able to consistently improve the test set perplexity. The test set WER shows a similar trend except for the last model with over 20 tuning parameters. By allowing independent tuning parameters across n-gram orders, we not only increase the risk of overfitting the development set data, but also significantly increase the model training time. Thus, we generally tie the GLI parameters across n-gram orders to improve robustness.

4.6 Summary

In this chapter, we presented a model-based approach to language model combination. The resulting generalized linear interpolation model defines a family of interpolation weight functions that subsumes both linear interpolation and count merging. Evaluations on the lecture transcription domain demonstrate up to a 0.8% and 0.3% WER drop over the baseline linear interpolation and count merging models, respectively. Detailed analysis shows that the generalized linear interpolation model outperforms both baselines with only 500 words of development set data.

In future work, we plan on studying the effectiveness of additional n-gram history features and different families of interpolation weight functions, including non-parametric and hierarchical approaches. We would also like to model the backoff weight as a function of n-gram features to improve the concentration of the interpolated probabilities.

As effective language modeling techniques for new domains with limited data increasingly rely on model combination approaches, the traditional linear interpolation and count merging techniques are no longer sufficient to capture and combine the essence of the constituent models. This chapter presents the generalized linear interpolation model as a step towards a principled study of model combination techniques. In the next chapter, we apply a similar feature-driven technique to de-emphasize out-of-domain n-grams during LM estimation. Instead of adjusting the interpolation weights for each n-gram context, we use the log-linear model to estimate the relevance of each n-gram to the target domain. By weighting the effective n-gram counts used during n-gram model estimation by their estimated relevance, we reduce the influence that out-of-domain n-grams have on the target model.

Chapter 5

n-gram Weighting

5.1 Motivation

Many application domains in machine learning suffer from a dearth of matched training data. However, partially matched data sets are often available in abundance. Past attempts to utilize the mismatched data for training often result in models that exhibit biases not observed in the target domain. In this chapter, we investigate the use of the often readily available data segmentation and metadata attributes associated with each corpus to reduce the effect of such bias. Specifically, we will examine this approach in the context of language modeling for lecture transcription.

Training corpora are often segmented into documents with associated metadata, such as title, date, and speaker. For lectures, if the data contains even a few lectures on linear algebra, conventional language modeling methods that lump the documents together will tend to assign disproportionately high probability to frequent terms like *vector* and *matrix*. Can we utilize the segmentation and metadata information to reduce the biases resulting from training data mismatch?

In this work, we present such a technique where we weight each n-gram count in a standard n-gram language model (LM) estimation procedure by a relevance factor computed from a log-linear combination of n-gram features. Utilizing features that correlate with the specificity of n-grams to subsets of the training documents, we effectively de-emphasize out-of-domain n-grams. By interpolating models, such as general lectures and course textbook, that match the target domain in complementary ways, and optimizing the weighting and interpolation parameters jointly, we allow each n-gram probability to be modeled by the most relevant interpolation component. Using a combination of features derived from multiple partitions of the training documents, the resulting weighted n-gram model achieves up to a 1.1% and 0.7% absolute word error rate (WER) reduction over the linear interpolation and count merging baselines on the lecture transcription task, respectively.

In this chapter, we assess the effectiveness of n-gram weighting towards reducing training data mismatch in cross-domain language model estimation scenarios. First, we describe the related work that partially motivated our approach. Next, we present the n-gram weighting technique and the features considered in this work. After evaluating the performance of n-gram weighting with various feature combinations against several smoothing and interpolation techniques, we examine the optimized model parameters and further characterize its behavior under different training conditions. Finally, we summarize the results and suggest ideas for future exploration.

5.2 Related Work

In Chapter 3, we introduced the use of the HMM-LDA [67] topic labels to reduce topic mismatch in LM estimation. Using an ad hoc method to reduce the effective counts of n-grams ending on topic words, we achieved better perplexity and WER than standard trigram LMs. Intuitively, de-emphasizing such n-grams will lower the transition probability to out-of-domain topic words from the training data. In this work, we further explore this intuition with a principled feature-based model, integrated with LM smoothing and interpolation to allow simultaneous optimization of all model parameters.

As Gao and Lee [46] observed, even purported matched training data may exhibit topic, style, or temporal biases not present in the test set. To address the mismatch, they partition the training documents by their metadata attributes and compute a measure of the likelihood that an n-gram will appear in a new partitioned segment. By pruning n-grams with generality probability below a given threshold, the resulting model achieves lower perplexity than a corresponding count-cutoff model of equal size. Complementary to our work, this technique also utilizes segmentation and metadata information. However, our model enables the simultaneous use of all metadata attributes by combining features derived from different partitions of the training documents within a single model.

5.3 n-gram Weighting

5.3.1 Model

Given a limited amount of training data, an n-gram appearing frequently in a single document may be assigned a disproportionately high probability. For example, an LM trained from lecture transcripts tends to assign excessive probability to words from observed lecture topics due to insufficient coverage of the underlying document topics. On the other hand, excessive probabilities may also be assigned to n-grams appearing consistently across documents with mismatched style, such as the course textbook in the written style. Traditional n-gram smoothing techniques do not address such issues of sparse topic coverage and style mismatch.

One approach to addressing the above issues is to weight the counts of the n-grams according to the concentration of their document distributions. Assigning higher weights to n-grams with evenly spread distributions captures the style of a data set, as reflected across all documents. On the other hand, emphasizing the n-grams concentrated within a few documents focuses the model on the topics of the individual documents.

In theory, n-gram weighting can be applied to any smoothing algorithm based on counts, such as Katz [84], Witten-Bell [152], and Kneser-Ney [90] smoothing. Because many of these algorithms require integer counts as input, we apply the weighting factors after the counts have been smoothed. For modified Kneser-Ney smoothing [22], applying n-gram weighting yields:

$$p(w|h) = \frac{\boldsymbol{\beta(hw)}\tilde{c}'(hw)}{\sum_w \boldsymbol{\beta(hw)}\tilde{c}(hw)} + \alpha(h)p(w|h')$$

where $p(w|h)$ is the probability of word w given history h, $\tilde{c}(wh)$ is the adjusted Kneser-Ney count, $\tilde{c}'(hw)$ is the discounted count, $\beta(hw)$ is the newly introduced n-gram weighting factor, $\alpha(h)$ is the normalizing backoff weight, and h' is the backoff history.

Although the weighting factor $\beta(hw)$ can in general be any function of the n-gram, in this work, we will consider a log-linear combination of n-gram features, or:

$$\beta(hw) = \exp(\mathbf{\Phi}(hw) \cdot \boldsymbol{\theta})$$

where $\mathbf{\Phi}(hw)$ is the feature vector for n-gram hw and $\boldsymbol{\theta}$ specifies the feature parameter vector to be learned. To better fit the data, we allow independent parameter vectors $\boldsymbol{\theta}_o$ for each n-gram order o. Note that with $\beta(hw) = 1$, the model degenerates to the original modified Kneser-Ney formulation. Furthermore, $\beta(hw)$ only specifies the relative weighting among n-grams with a common history h. Thus, scaling $\beta(hw)$ by an arbitrary function $f(h)$ has no effect on the model.

In isolation, n-gram weighting shifts probability mass from out-of-domain n-grams via backoff to the uniform distribution to improve the generality of the resulting model. However, in combination with LM interpolation, it can also distribute probabilities to LM components that better model specific n-grams. For example, n-gram weighting can de-emphasize off-topic and off-style n-grams from general lectures and course textbook, respectively. Tuning the weighting and interpolation parameters jointly further allows the estimation of the n-gram probabilities to utilize the best matching LM components.

5.3.2 Features

To address the issue of sparsity in the document topic distribution, we can apply n-gram weighting with features that measure the concentration of the n-gram distribution across documents. Similar features can also be computed from documents partitioned by their categorical metadata attributes, such as *course* and *speaker* for lecture transcripts. Whereas the features derived from the corpus documents should correlate with the topic specificity of the n-grams, the same features computed from the speaker partitions might correspond to the speaker specificity. By combining features from multiple partitions of the training data to compute the weighting factors, n-gram weighting allows the resulting model to better generalize across categories.

To guide the presentation of the n-gram features below, we will consider the following example partition of the training corpus. Words tagged by HMM-LDA as topic words appear in bold.

A B A	A **C C** A	**B** A B
B A **A** C	C B A	A B A
A C B	A A **C**	A **B B** A

One way to estimate the specificity of an n-gram across partitions is to measure the n-gram frequency $F(hw)$, or the fraction of partitions containing the n-gram hw. For instance, $F(\text{A}) = 3/3$, $f(\text{C}) = 2/3$. However, as the size of each partition increases, this ratio tends to increase towards 1, since most n-grams have a non-zero probability of appearing in each partition. Thus, an alternative is to compute the normalized entropy of the n-gram distribution across the S partitions, or $H(hw) = \frac{-1}{\log S} \sum_{s=1}^{S} p_s(hw) \log p_s(hw)$, where $p_s(hw)$ is the fraction of the n-gram hw that are observed in partition s. Unseen n-grams are assigned the value 0. For example, the normalized entropy of the unigram C is $H(\text{C}) = \frac{-1}{\log 3}[\frac{2}{6} \log \frac{2}{6} + \frac{4}{6} \log \frac{4}{6} + 0] = 0.58$. n-grams clustered in fewer partitions have lower entropy than ones that are more evenly spread out.

Feature	of the	i think	k means	the sun	this is a	a lot of	big o of	e m f
Random	0.03	0.32	0.33	0.19	0.53	0.24	0.37	0.80
$\log(c(hw))$	9.29	8.09	3.47	5.86	6.82	7.16	3.09	4.92
$F^{\text{doc}}(hw)$	1.00	0.93	0.00	0.18	0.92	0.76	0.00	0.04
$F^{\text{course}}(hw)$	1.00	1.00	0.06	0.56	0.94	0.94	0.06	0.06
$F^{\text{speaker}}(hw)$	0.83	0.70	0.00	0.06	0.41	0.55	0.01	0.00
$H^{\text{doc}}(hw)$	0.96	0.84	0.00	0.56	0.93	0.85	0.00	0.34
$H^{\text{course}}(hw)$	0.75	0.61	0.00	0.55	0.78	0.65	0.00	0.00
$H^{\text{speaker}}(hw)$	0.76	0.81	0.00	0.09	0.65	0.80	0.12	0.00
$T^{\text{doc}}(hw)$	0.00	0.00	0.91	1.00	0.01	0.00	0.00	0.04
$T^{\text{course}}(hw)$	0.00	0.00	0.88	0.28	0.01	0.00	0.00	1.00
$T^{\text{speaker}}(hw)$	0.00	0.00	0.94	0.92	0.01	0.00	0.09	0.99

Table 5.1: A list of n-gram weighting features and their values. c: count, F: n-gram frequency, H: normalized entropy, T: topic probability.

Following Section 3.5.1, we also consider features derived from the HMM-LDA word topic labels.[1] Specifically, we compute the empirical probability $T(hw)$ that the target word w of the n-gram hw is labeled as a topic word. The probabilities of unobserved n-grams are assigned the value 0. In the example corpus, $T(\text{C}) = 3/6$, $T(\text{A C}) = 2/4$.

All of the above features can be computed for any partitioning of the training data. To better illustrate the differences, we compute the features on the Lectures corpus, partitioned by lecture (doc), course, and speaker. Furthermore, we include the log of the n-gram counts $c(hw)$ and random values between 0 and 1 as baseline features. Table 5.1 lists all the features examined in this work and their values on a select subset of n-grams.

5.3.3 Training

In Chapter 7, we present an efficient n-gram data structure and optimized algorithms specifically designed for iterative language model estimation. With such an implementation, we can jointly tune the n-gram weighting and interpolation parameters using any numerical optimization technique [123] to iteratively estimate the optimal parameter values.

In this chapter, we apply Powell's method [123] to numerically minimize the development set perplexity, which iteratively optimizes the objective function along a set of continually updated directions using single-variable golden-ratio line search without requiring computation of the function gradient. Although there is no guarantee against converging to a local minimum when jointly tuning both the n-gram weighting and interpolation parameters, we have found that initializing the n-gram weighting parameters to zero generally yields good performance.

[1] HMM-LDA is performed using 20 states and 50 topics with a 3rd-order HMM. Hyperparameters are sampled with a log-normal Metropolis proposal. The model with the highest likelihood from among 10,000 iterations of Gibbs sampling is used.

	Perplexity		WER	
Model	Dev	Test	Dev	Test
FixKN(1)	174.7	196.7	34.7%	36.7%
+ W(H^{doc})	172.9	194.8	34.7%	36.6%
FixKN(3)	168.6	189.3	34.8%	36.8%
+ W(H^{doc})	166.8	187.8	34.6%	36.6%
FixKN(10)	167.5	187.6	35.0%	37.0%
+ W(H^{doc})	165.3	185.8	34.7%	36.7%
KN(1)	169.7	190.4	34.9%	36.7%
+ W(H^{doc})	167.3	188.2	34.7%	36.5%
KN(3)	163.4	183.1	35.0%	36.8%
+ W(H^{doc})	161.1	181.2	34.8%	36.5%
KN(10)	162.3	181.8	35.0%	36.8%
+ W(H^{doc})	160.1	180.0	34.8%	36.5%

Table 5.2: Performance of n-gram weighting with a variety of Kneser-Ney settings. FixKN(d): Kneser-Ney with d fixed discount parameters. KN(d): FixKN(d) with tuned values. W(*feat*): n-gram weighting with *feat* feature.

5.4 Experiments

5.4.1 Setup

In this work, we compute the perplexity and WER of various trigram LMs with the default corpus vocabulary on the lecture transcription task (see Section 2.2.1) to evaluate the effectiveness of n-gram weighting under multiple conditions. Since all the models considered in this work can be encoded as ARPA n-gram backoff models, they can be applied directly during the first recognition pass. However, to reduce the computation time, we report in these experiments the WER performance obtained by rescoring the recognition lattices generated from the count-merging LM without any n-gram weighting, unless otherwise noted. The WER difference resulting from the use of lattice rescoring is about 0.01%.

5.4.2 Smoothing

In Table 5.2, we compare the performance of n-gram weighting with the H^{doc} document entropy feature for various modified Kneser-Ney smoothing configurations [22] on the Lectures dataset. Specifically, we consider varying the number of discount parameters per n-gram order from 1 to 10. The original and modified Kneser-Ney smoothing algorithms correspond to a setting of 1 and 3, respectively. Furthermore, we explore using both fixed parameter values estimated from n-gram count statistics and tuned values that minimize the development set perplexity.

In this task, while the test set perplexity tracks the development set perplexity well, the WER correlates surprisingly poorly with the perplexity on both the development and test sets. Nevertheless, n-gram weighting consistently reduces the absolute test set WER by a statistically significant average of 0.3%, according to the Matched Pairs Sentence Segment Word Error test [118]. Given that we obtained the lowest development set WER with the fixed 3-parameter modified Kneser-Ney smoothing, all subsequent experiments are conducted using this smoothing configuration.

Model	Perplexity		Word Error Rate	
	Development	Test	Development	Test
Lectures	189.3	189.3	34.8%	36.8%
+ W(H^{doc})	166.8 (-1.1%)	187.8 (-0.8%)	34.5%	36.5%
Textbook	283.4	326.1	41.9%	42.4%
+ W(H^{doc})	277.0 (-2.3%)	317.5 (-2.6%)	42.0%	42.5%
LI(Lectures + Textbook)	125.6	141.6	31.7%	33.4%
+ W(H^{doc})	121.5 (-3.3%)	136.6 (-3.5%)	31.0%	32.5%

Table 5.3: n-gram weighting with linear interpolation.

5.4.3 Linear Interpolation

Applied to the Lectures dataset in isolation, n-gram weighting with the H^{doc} feature reduces the test set WER by 0.3% by de-emphasizing the probability contributions from off-topic n-grams and shifting their weights to the backoff distributions. Ideally though, such weights should be distributed to on-topic n-grams, perhaps from other LM components.

In Table 5.3, we present the performance of applying n-gram weighting to the Lectures and Textbook models individually versus in combination via linear interpolation (LI), where we optimize the n-gram weighting and interpolation parameters jointly. The interpolated model with n-gram weighting achieves perplexity improvements roughly additive of the reductions obtained with the individual models. However, the 0.9% WER drop for the interpolated model significantly exceeds the sum of the individual reductions. Thus, as we will examine in more detail in Section 5.5.1, n-gram weighting allows probabilities to be shifted from less relevant n-grams in one component to more specific n-grams in another.

5.4.4 Features

With n-gram weighting, we can model the weighting function $\beta(hw)$ as a log-linear combination of any n-gram features. In Table 5.4, we show the effect various features have on the performance of linearly interpolating Lectures and Textbook. As the documents from the Lectures dataset is annotated with course and speaker metadata attributes, we include the n-gram frequency F, normalized entropy H, and topic probability T features computed from the lectures grouped by the 16 unique courses and 299 unique speakers.[2]

In terms of perplexity, the use of the *Random* feature has negligible impact on the test set performance, as expected. On the other hand, the $\log(c)$ count feature reduces the perplexity by nearly 3%, as it correlates with the generality of the n-grams. By using features that leverage the information from document segmentation and associated metadata, we are generally able to achieve greater perplexity reductions. Overall, the frequency and entropy features perform roughly equally. However, by considering information from the more sophisticated HMM-LDA topic model, the topic probability feature T^{doc} achieves significantly lower perplexity than any other feature in isolation.

In terms of WER, the *Random* feature again shows no effect on the baseline WER of 33.4%. However, to our surprise, the use of the simple $\log(c)$ feature achieves nearly the same

[2]Features that are not applicable to a particular corpus (e.g. H^{course} for Textbook) are removed from the n-gram weighting computation for that component. Thus, models with course and speaker features have fewer tunable parameters than the others.

Model	Perplexity		Word Error Rate	
	Development	Test	Development	Test
LI(Lectures + Textbook)	125.6	141.6	31.7%	33.4%
+ W(*Random*)	125.6 (-0.0%)	141.5 (-0.1%)	31.7%	33.4%
+ W($\log(c)$)	122.2 (-2.7%)	137.5 (-2.9%)	*31.1%*	*32.6%*
+ W(F^{doc})	121.4 (-3.3%)	136.3 (-3.7%)	*31.2%*	*32.6%*
+ W(F^{course})	121.3 (-3.4%)	136.5 (-3.6%)	*31.0%*	*32.5%*
+ W(F^{speaker})	122.7 (-2.3%)	138.1 (-2.5%)	*31.2%*	*32.8%*
+ W(H^{doc})	121.5 (-3.3%)	136.6 (-3.5%)	*31.0%*	***32.5%***
+ W(H^{course}) ★	121.2 (-3.5%)	136.1 (-3.9%)	***30.9%***	*32.5%*
+ W(H^{speaker})	123.4 (-1.8%)	138.6 (-2.1%)	*31.3%*	*32.9%*
+ W(T^{doc})	**120.9** (-3.8%)	**134.8** (-4.8%)	*31.4%*	*32.9%*
+ W(T^{course})	121.7 (-3.1%)	136.4 (-3.6%)	*31.4%*	*32.9%*
+ W(T^{speaker})	121.7 (-3.2%)	136.4 (-3.7%)	*31.4%*	*32.9%*

Table 5.4: n-gram weighting with various features. Statistically significant improvements ($p < 0.05$) appear in italics.

WER improvement as the best segmentation-based feature, whereas the more sophisticated features computed from the HMM-LDA labels only obtain half of the reduction even though they have the best perplexities.

When comparing the performance of different n-gram weighting features on this data set, the perplexity correlates poorly with the WER, on both the development and test sets. Fortunately, the features that yield the lowest perplexity and WER on the development set also yield one of the lowest perplexities and WERs, respectively, on the test set. Thus, during feature selection for speech recognition applications, we should consider the development set WER. Specifically, since the differences in WER are often statistically insignificant, we will select the feature that minimizes the sum of the development set WER and log perplexity, or cross-entropy.[3] In Tables 5.4 and 5.5, we have bolded the perplexities and WERs of the features with the lowest values. The features that achieve the lowest combined cross-entropy and WER on the development set are starred.

5.4.5 Feature Combination

Unlike most previous work, n-gram weighting enables a systematic integration of features computed from multiple document partitions. In Table 5.5, we compare the performance of various feature combinations. We experiment with incrementally adding features that yield the lowest combined development set cross-entropy and WER. Overall, this metric appears to better predict the test set WER than either the development set perplexity or WER alone.

Using the combined feature selection technique, we notice that the greedily selected features tend to differ in the choice of document segmentation and feature type, suggesting that n-gram weighting can effectively integrate the information provided by the document metadata. By combining features, we are able to further reduce the test set WER by a statistically significant ($p < 0.001$) 0.2% over the best single feature model.

[3]The choice of cross-entropy instead of perplexity is partially motivated by the linear correlation reported by Chen and Goodman [22] between cross-entropy and WER.

Features	Perplexity		Word Error Rate	
	Development	Test	Development	Test
H^{course}	121.2	136.1	30.9%	32.5%
+ $\log(c)$	120.7 (-0.4%)	135.4 (-0.5%)	30.9%	*32.5%*
+ F^{doc}	120.6 (-0.5%)	135.1 (-0.7%)	30.9%	***32.4%***
+ H^{doc}	120.8 (-0.3%)	135.6 (-0.4%)	30.9%	32.5%
+ T^{doc} ★	**119.5** (-1.4%)	**133.2** (-2.1%)	30.9%	32.4%
+ F^{course}	121.1 (-0.1%)	136.0 (-0.1%)	*31.0%*	*32.4%*
+ T^{course}	120.2 (-0.8%)	134.8 (-1.0%)	30.9%	32.5%
+ F^{speaker}	121.1 (-0.1%)	136.0 (-0.1%)	30.9%	*32.5%*
+ H^{speaker}	121.1 (-0.1%)	136.1 (-0.0%)	30.9%	32.6%
+ T^{speaker}	120.2 (-0.8%)	134.7 (-1.0%)	**30.8%**	32.4%
H^{course} + T^{doc}	119.5	133.2	30.9%	32.4%
+ $\log(c)$ ★	**119.2** (-0.3%)	132.8 (-0.3%)	***30.8%***	***32.3%***
+ F^{doc}	119.2 (-0.3%)	**132.8** (-0.4%)	30.9%	32.4%
+ H^{doc}	119.3 (-0.2%)	133.0 (-0.2%)	*30.8%*	32.4%
+ F^{course}	119.4 (-0.1%)	133.1 (-0.1%)	30.9%	32.4%
+ T^{course}	119.4 (-0.1%)	133.0 (-0.1%)	31.0%	32.4%
+ F^{speaker}	119.4 (-0.1%)	133.1 (-0.1%)	30.8%	32.4%
+ H^{speaker}	119.4 (-0.1%)	133.2 (-0.0%)	31.0%	32.4%
+ T^{speaker}	119.4 (-0.1%)	133.1 (-0.1%)	30.9%	32.4%

Table 5.5: n-gram weighting with feature combinations. Statistically significant improvements ($p < 0.05$) appear in italics.

5.4.6 Advanced Interpolation

While n-gram weighting with all three features $\mathbf{\Phi} = [H^{\text{course}}, T^{\text{doc}}, \log(c)]$ is able to reduce the test set WER by 1.1% over the linear interpolation baseline, linear interpolation is not a particularly effective interpolation technique. In Table 5.6, we compare the effectiveness of n-gram weighting in combination with better interpolation techniques, such as count merging (CM) [1] and generalized linear interpolation (GLI) (see Chapter 4).

As expected, the use of more sophisticated interpolation techniques decreases the perplexity and WER reductions achieved via n-gram weighting by roughly half for a variety of feature combinations. However, all improvements remain statistically significant. Although we initially achieve a significant WER reduction of 0.8% from the use of generalized linear interpolation over simple linear interpolation, as we add n-gram weighting features, the benefit of more sophisticated interpolation techniques decreases rapidly. With all three n-gram weighting features, the test set WER difference, although still statistically significant, is reduced to only 0.2%. Unfortunately, the improvements obtained from n-gram weighting and generalized linear interpolation do not appear to be additive.

Model	# Params	Perplexity		Word Error Rate	
		Development	Test	Development	Test
LI(L + T)	1	125.6	141.6	31.7%	33.4%
+ W(H^{course})	4	121.2 (-3.5%)	136.1 (-3.9%)	30.9%	32.5%
+ W(T^{doc})	10	119.5 (-4.9%)	133.2 (-5.9%)	30.9%	32.4%
+ W(log(c))	16	119.2 (-5.2%)	132.8 (-6.2%)	30.8%	32.3%
CM(L + T)	1	122.3	138.1	31.4%	32.9%
+ W(H^{course})	4	119.9 (-2.0%)	135.0 (-2.2%)	30.9%	32.3%
+ W(T^{doc})	10	118.5 (-3.1%)	132.7 (-3.9%)	30.9%	32.2%
+ W(log(c))	16	117.8 (-3.6%)	131.9 (-4.5%)	30.9%	32.2%
GLI(L + T)	7	119.7	135.1	31.0%	32.6%
+ W(H^{course})	10	117.5 (-1.8%)	132.5 (-2.0%)	30.5%	32.2%
+ W(T^{doc})	16	116.0 (-3.1%)	129.9 (-3.8%)	30.4%	32.1%
+ W(log(c))	22	115.5 (-3.6%)	129.3 (-4.3%)	30.5%	32.1%

Table 5.6: Effect of interpolation technique. Generalized linear interpolation is trained with features $\mathbf{\Psi}(h) = [\log(\sum_w \tilde{c}(hw)), T^w(h), F(h)]$ (see Section 4.3.3). L: Lectures, T: Textbook.

Feature	Parameter Values		
	$\boldsymbol{\theta}^{\mathsf{L}}$	$\boldsymbol{\theta}^{\mathsf{T}}$	$[\lambda^{\mathsf{L}}, \lambda^{\mathsf{T}}]$
H^{doc}	$[3.42, 1.46, 0.12]$	$[-0.45, -0.35, -0.73]$	$[0.67, 0.33]$
T^{doc}	$[-2.33, -1.63, -1.19]$	$[1.05, 0.46, 0.12]$	$[0.68, 0.32]$

Table 5.7: n-gram weighting parameter values. $\boldsymbol{\theta}^{\mathsf{L}}$, $\boldsymbol{\theta}^{\mathsf{T}}$: parameters for each order of the Lectures and Textbook trigram models, λ^{L},λ^{T}: linear interpolation weights.

5.5 Analysis

5.5.1 Weighting Parameters

To obtain further insight into how n-gram weighting improves the resulting n-gram model, we present in Table 5.7 the optimized parameter values for the linear interpolation model between Lectures and Textbook using n-gram weighting with H^{doc} and T^{doc} features. Using $\beta(hw) = \exp(\mathbf{\Phi}(hw) \cdot \boldsymbol{\theta})$ to model the n-gram weights, a positive value of θ_i corresponds to increasing the weights of the i^{th} order n-grams with positive feature values.

For the H^{doc} normalized entropy feature, values close to 1 correspond to n-grams that are evenly distributed across the documents. When interpolating Lectures and Textbook, we obtain consistently positive values for the Lectures component, indicating a de-emphasis on document-specific terms that are unlikely to be found in the target computer science domain. On the other hand, the values corresponding to the Textbook component are consistently negative, suggesting a reduced weight for mismatched style terms that appear uniformly across textbook sections.

For T^{doc}, values close to 1 correspond to n-grams ending frequently on topic words with uneven distribution across documents. Thus, as expected, the signs of the optimized parameter values are flipped. By de-emphasizing topic n-grams from off-topic components and style n-grams from off-style components, n-gram weighting effectively improves the performance of the resulting language model.

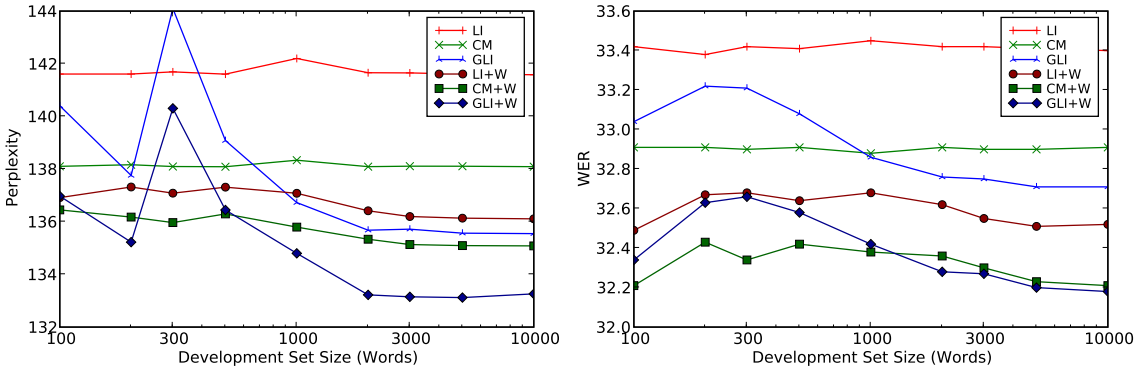


Figure 5-1: Test set performance vs. development set size. n-gram weighting uses features $\mathbf{\Phi} = [H^{\text{course}}]$. GLI uses features $\mathbf{\Psi}(h) = [\log(\sum_w \tilde{c}(hw))]$. Lattice rescoring is not used.

5.5.2 Development Set Size

So far, we have assumed the availability of a large development set for parameter tuning. To obtain a sense of how n-gram weighting performs with smaller development sets, we randomly select utterances from the full development set and plot the test set performance in Figure 5-1 as a function of the development set size for various modeling techniques.

As expected, GLI with features $\mathbf{\Psi}(h) = [\log(\sum_w \tilde{c}(hw))]$ outperforms both LI and CM. However, whereas LI and CM essentially converge in test set perplexity with only 100 words of development data, it takes about 2,000 words before GLI converges due to the increased number of parameters. By adding n-gram weighting with the H^{course} feature, we see a significant drop in perplexity for all models at all development set sizes. However, the performance does not fully converge until 3,000 words of development set data.

In terms of WER, n-gram weighting decreases the WER significantly across all development set sizes and interpolation techniques. As GLI has 3 more tuning parameters than CM, it does not outperform CM until the development set size increases to 2,000 words. From these results, we conclude that although n-gram weighting increases the number of tuning parameters, they are effective in improving the test set performance even with only 100 words of development set data.

5.5.3 Training Set Size

To characterize the effectiveness of n-gram weighting as a function of the training set size, we evaluate the performance of various interpolated models with increasing subsets of the Lectures corpus and the full Textbook corpus. Overall, every doubling of the number of training set documents decreases both the test set perplexity and WER by approximately 7 points and 0.8%, respectively. To better compare results, we plot the performance difference between various models and linear interpolation in Figure 5-2.

Interestingly, the peak gain obtained from n-gram weighting with the H^{doc} feature appears at around 16 documents for all interpolation techniques. We suspect that as the number of documents initially increases, the estimation of the H^{doc} features improves, resulting in larger perplexity reduction from n-gram weighting. However, as the diversity of the training set documents increases beyond a certain threshold, we experience less document-level sparsity. Thus, we see decreasing gains from n-gram weighting beyond 16 documents.

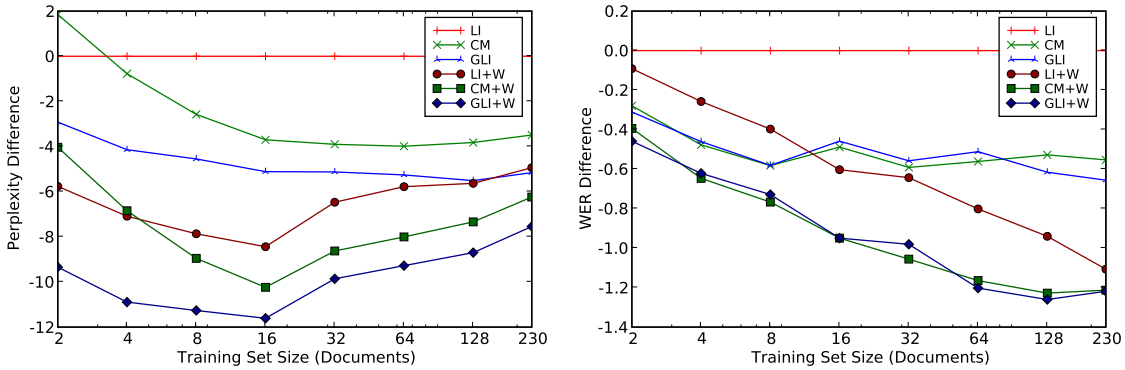


Figure 5-2: Test set performance vs. training set size. n-gram weighting uses features $\mathbf{\Phi} = [H^{\text{doc}}]$. GLI uses features $\mathbf{\Psi}(h) = [\log_{+1}(\sum_w \tilde{c}(hw))]$. Lattice rescoring is not used.

	Perplexity / WER			
Model	T + L	T + S	L + S	T + L + S
LI	141.6 / 33.4%	169.4 / 35.8%	191.6 / 36.5%	139.5 / 33.3%
LI + W	132.8 / 32.3%	161.6 / 35.0%	184.5 / 36.4%	130.9 / 32.2%
CM	138.1 / 32.9%	171.4 / 35.3%	194.2 / 36.5%	136.9 / 32.9%
CM + W	131.9 / 32.2%	164.0 / 34.9%	186.9 / 36.3%	130.4 / 32.3%
GLI	135.1 / 32.6%	166.7 / 35.0%	190.8 / 36.5%	132.5 / 32.6%
GLI + W	129.3 / 32.1%	159.7 / 34.5%	181.1 / 36.1%	125.7 / 31.9%

Table 5.8: Test set performance with various corpus combinations. L: Lectures, T: Textbook, S: Switchboard, W: n-gram weighting.

For all interpolation techniques, even though the perplexity improvements from n-gram weighting decrease with more documents, the WER reductions actually increase. Furthermore, n-gram weighting shows statistically significant reductions for all configurations except linear interpolation with 2 documents. Thus, even with minimal segmentation, n-gram weighting with segment-based features is effective in improving the test set performance.

5.5.4 Training Corpora

In Table 5.8, we compare the performance of n-gram weighting with different combinations of training corpora and interpolation techniques to determine its effectiveness across different training conditions. All model combinations yield statistically significant improvements with n-gram weighting using features $\mathbf{\Phi}(hw) = [H^{\text{course}}(hw), T^{\text{doc}}(hw), \log(c(hw))]$.

The results suggest that n-gram weighting with these features is most effective when the interpolation corpora differ in how they match the target domain. Whereas the Textbook corpus is the only corpus with matching topic, both Lectures and Switchboard have a similar matching spoken conversational style. Thus, we see the least benefit from n-gram weighting when interpolating Lectures and Switchboard. By combining Lectures, Textbook, and Switchboard using generalized linear interpolation with n-gram weighting, we achieve our best test set WER of 31.9% on the lecture transcription task, a full 1.5% over the initial linear interpolation baseline.

5.6 Summary

In this chapter, we presented the n-gram weighting technique for adjusting the probabilities of n-grams according to a set of features. By utilizing features derived from the document segmentation and associated metadata inherent in many training corpora, we achieved up to a 1.1% and 0.7% WER reduction over the linear interpolation and count merging baselines, respectively, using n-gram weighting on the lecture transcription task.

We examined the performance of various n-gram weighting features and generally found entropy-based features to offer the best predictive performance. Although the topic probability features derived from HMM-LDA labels yield additional improvements when applied in combination with the normalized entropy features, the computational cost of performing HMM-LDA may not justify the marginal benefit in all scenarios.

In situations where the document boundaries are unavailable or when finer segmentation is desired, automatic techniques for document segmentation may be applied [103]. Synthetic metadata information may also be obtained via clustering techniques [137]. Although we have primarily focused on n-gram weighting features derived from segmentation information, it is also possible to consider other features that correlate with n-gram relevance.

n-gram weighting and other approaches to cross-domain language modeling require a matched development set for model parameter tuning. Thus, in the next chapter, we investigate the use of the initial recognition hypotheses as the development set, as well as manually transcribing a subset of the test set utterances. We follow up in Chapter 7 with a description of an efficient implementation of n-gram weighting and other language modeling techniques involving iterative parameter estimation.

Chapter 6

Parameter Optimization

6.1 Motivation

In an ideal world, language model (LM) parameters would be learned on training data that exactly matches testing conditions. In practice however, many potentially valuable applications for speech processing have only limited amounts of matched training data. In extreme cases, no such data exists beyond the unlabeled test data itself. As described in Section 2.2, one example of this latter situation are recordings of academic lectures. While we may have existing transcripts from general lectures or written text on the precise topic, data that matches both the topic and style of the target lecture and speaker rarely exist.

In previous chapters, we introduced various language modeling techniques, including generalized linear interpolation (Chapter 4) and n-gram weighting (Chapter 5), that take advantage of partially matched training data to more effectively model the target domain. There, we assumed the availability of a large in-domain development set that perfectly matches the target lecture in both topic and style to tune the interpolation and weighting parameters. In this chapter, we will examine alternative techniques for parameter tuning via unsupervised and active learning approaches.

In off-line lecture transcription scenarios, we can perform multiple recognition passes over the lecture. Thus, the recognition hypotheses from previous iterations may be used as data approximating the target domain. In addition, since users are often motivated to obtain the most accurate transcription possible, they may be willing to transcribe a subset of the lecture utterances to aid the recognition. The resulting user transcriptions not only serve as in-domain development data, but also eliminate the effect of any recognition errors among these utterances.

In this chapter, we explore various supervised and unsupervised approaches to tune the model parameters, with a focus on language modeling for lecture transcription. We first evaluate the effectiveness of using the recognition hypotheses as development data. Next, we study the performance of using various amounts of user transcription for parameter tuning. As the system can select the order in which utterances are presented to the user for transcription, we compare the effectiveness of transcribing utterances by chronological order, random sampling, and lowest utterance confidence. We apply the above techniques to the LM interpolation and weighting schemes presented in this thesis. With only 300 words of user transcription, we were able to outperform unsupervised adaptation using the recognition hypotheses. By transcribing the utterances in increasing confidence, we further reduce the effective word error rate (WER) by 0.6%.

6.2 Related Work

Previous language modeling research that deals with sparse training data has investigated various adaptation and interpolation techniques that make use of partially matched corpora [6, 70, 74]. However, they generally assume the existence of an independent in-domain development set for parameter tuning. Several researchers have explored using the recognition hypotheses for language model adaptation. Typically, the hypotheses are used to build a component LM to be combined with the baseline LM via linear interpolation [111, 144] or count merging [2]. However, in certain settings, minimum discrimination information adaptation has been shown to yield slightly better results [20, 115]. Since the recognition hypotheses change through adaptation, the process can be repeated iteratively for potentially further improvements [2, 115].

In existing work, model adaptation parameters are often chosen arbitrarily. With only one parameter that specifies the weight of the adaptation data with respect to the baseline LM, tuning the parameter on an accurate development set is not always critical [144]. However, when the baseline model is itself interpolated and contains multiple parameters, having an error-free development set to optimize both the baseline and adaptation parameters becomes paramount, as the parameters are now more likely to fit the errors. Thus, in this work, we extend previous work by considering interpolated LM baselines with more sophisticated modeling techniques, such as count merging and n-gram weighting.

To obtain accurate in-domain data for tuning, we propose selecting utterances from the target lecture for transcription. Although active learning techniques have been proposed for selecting training utterances based on confidence scores [127], the selection criteria in this case need to balance between eliminating errors from the transcribed utterances and building a representative development set. In this chapter, we will measure the effect of such a tradeoff using multiple utterance selection techniques.

6.3 Experiments

In this work, we evaluate the perplexity and WER of various trigram LMs with the default corpus vocabulary on the lecture transcription task (see Section 2.2.1). The model parameters are tuned to minimize the perplexity of the recognition hypotheses and user transcriptions. Since all the models considered in this work can be encoded as ARPA n-gram backoff models, they can be applied directly during the first recognition pass. However, to reduce the computation time, we report in these experiments the WER performance obtained by rescoring the recognition lattices generated from the untuned count-merging LM without n-gram weighting. The WER difference resulting from the use of lattice rescoring is about 0.01%. We evaluate each of the 10 target lectures in CS Test independently and present the averages of the WER results.

In the following sections, we evaluate the effectiveness of using the recognition hypotheses and user transcriptions for parameter estimation on a variety of LM estimation and interpolation schemes. As a baseline, we consider trigram models smoothed using fixed-parameter modified Kneser-Ney [22] smoothing (KN). For better performance, we apply n-gram weighting with document entropy features (see Chapter 5) to the component models to de-emphasize out-of-domain n-grams (KN+W). We interpolate these component models built from Textbook and Lectures using linear interpolation (LI) and count merging (CM). Overall, the KN and KN+W model configurations have 1 and 7 parameters, respectively.

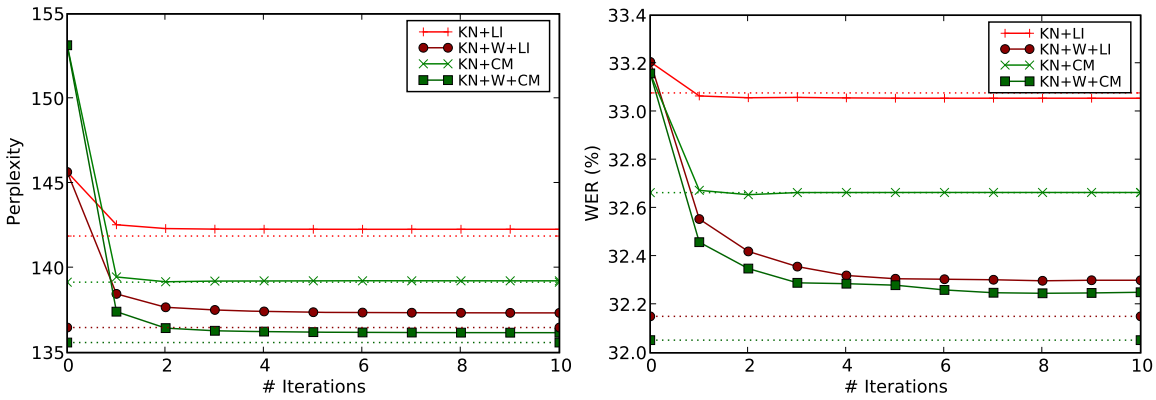


Figure 6-1: Average test set performance vs. adaptation iteration. The 0^{th} iteration corresponds to the unadapted model. The dotted lines correspond to the oracle performance obtained by tuning the model on the reference transcript.

6.3.1 Recognition Hypotheses

In scenarios where no matched training data is available, we can perform multiple recognition passes and use the 1-best recognition hypotheses from the previous pass as the development set for tuning the LM parameters. In Figure 6-1, we plot the performance for various LM configurations over 10 such iterations. The 0^{th} iteration corresponds to estimating the LM using the default parameter values. As baselines, we also include, as dotted lines, the oracle WERs obtained by tuning the model parameters directly on the reference transcript.

As observed in previous works, count merging outperforms linear interpolation. Without n-gram weighting, the WERs for both interpolation techniques converge in a single iteration. The errors in the recognition hypotheses appear to have negligible effect on the tuned performance of these 1-parameter models.[1]

Overall, applying n-gram weighting to the individual LM components significantly improves the performance of the resulting models by introducing additional tuning parameters. Unlike the 1-parameter models, the WER of the n-gram weighted models does not converge until about the 4^{th} iteration, with the first iteration achieving only about 75% of the total reduction. Furthermore, it is much more sensitive to errors in the development set, with a gap of 0.2% between the best unsupervised and the oracle WERs at about 30% WER.

In some configurations, unsupervised adaptation has previously been observed to degrade performance after a few iterations due to the reinforcement of previous recognition errors [2]. By using the recognition hypotheses only to tune the model parameters, the above models appear to be immune to such overfitting.

6.3.2 User-Transcription

Although human transcription of the entire lecture is often impractical, content providers and end-users are often motivated to help improve the recognition accuracy. Instead of using inaccurate recognition hypotheses, we may be able to obtain transcripts of select utterances from the users for the development set. Unlike general transcriptionists, users

[1]The difference between the best unsupervised and the oracle WERs for KN+LI is not statistically significant.

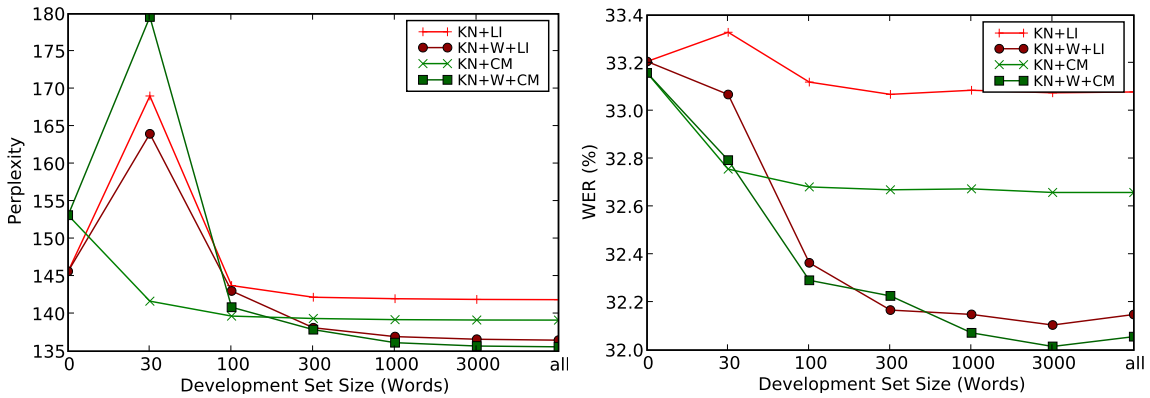


Figure 6-2: Average test set performance vs. development set size for various LM configurations. 0 corresponds to the unadapted model. all corresponds to approximately 10,000 words.

of the 'application are also more likely to correctly transcribe the technical jargon found in the target lectures (instead of *Markov* → *mark of* and *Fourier* → *for your*). In Figure 6-2, we plot the performance of various LM configurations trained with increasing amounts of development set data. Specifically, we incrementally add the reference transcripts of randomly selected utterances from the target lecture until we have obtained the desired minimum number of words.

As expected, increasing the development set size improves the LM performance. For simple linear interpolation and count merging, the WER converges after only 100 words. Although applying n-gram weighting reduces the WER by up to 1.0% absolute, it takes about 10 times more development set data before we observe convergence.

Compared with iterative unsupervised adaptation using the recognition hypotheses, we are able to achieve better recognition performance with only 300 words of transcribed development set data, or 3% of the target lecture, for all model configurations. With appropriate transcription tools that support re-dictation of the target utterance, easy correction of speech recognition errors [78], and a streamlined text entry interface [92], we expect most users to be able to transcribe 300 words within 15 minutes.

6.3.3 Utterance Selection

Since the applications often have the opportunity to pre-process the target lecture prior to presenting the utterances to the user for transcription, we can consider various utterance selection strategies to best utilize the user transcriptions. The most natural approach from the user perspective is to transcribe the utterances in chronological order, as they are easier to comprehend in context. However, given the topic shifts that frequently occur within a lecture (cf. Section 3.5.5), random selection may result in a development set that better represents the target lecture.

As user-transcribed utterances from the target lecture do not need to be processed by the speech recognizer, which adds to the overall WER, another strategy is to select utterances that are most likely to yield recognition errors. In our implementation, we compute the confidence of an utterance as the average of the 1-best word posterior probabilities in a normalized word lattice [127] and present the utterances in order of increasing confidence.

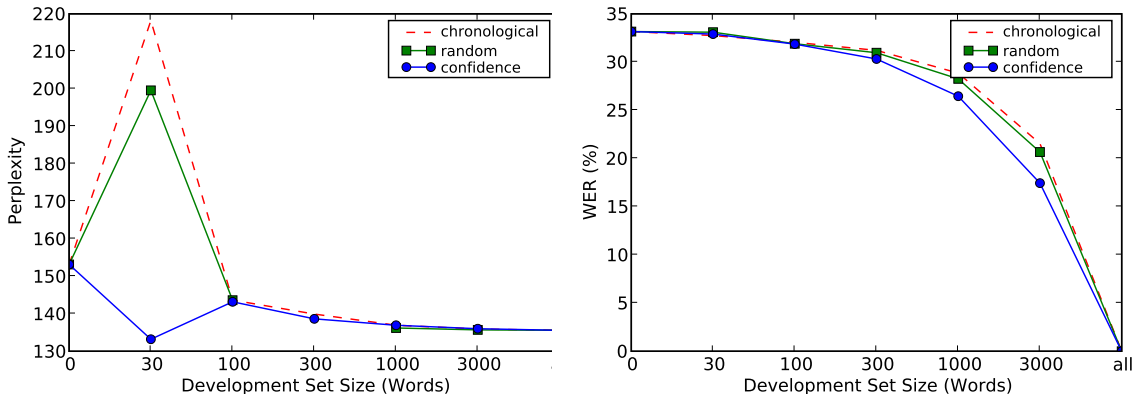


Figure 6-3: Average effective test set performance vs. development set size for the KN+W+CM configuration using various utterance selection strategies.

Since transcribed utterances do not contribute to the WER, we need to remove errors corresponding to transcribed utterances when computing the effective WER. Thus, when all utterances are transcribed, the effective WER is 0. In Figure 6-3, we plot the perplexity and effective WER against the minimum number of words transcribed using the chronological, random (averaged over 3 trials), and lowest confidence utterance selection strategies.

As predicted, random selection generally yields lower WER than chronological selection. Although transcribing utterances in order of lowest confidence may result in a development set less representative of the overall lecture, in this task, the benefit of not accumulating errors from these low confidence utterances outweigh the effects of the mismatch. If users can transcribe low confidence utterances with the same effort as high confidence ones, utterances to be transcribed should be selected in order of increasing confidence to minimize the effective WER.

6.4 Summary

In this chapter, we demonstrated the effectiveness of various techniques for tuning multi-parameter LMs when matched training data is unavailable. When interpolating Textbook and Lectures using count merging with n-gram weighting, optimizing the parameters using the recognition hypotheses as a development set achieves around 80% of the 1.1% oracle WER reduction obtained with the reference transcript. Whereas single-parameter models generally converge in a single adaptation iteration, more sophisticated models often require iterative adaptation to achieve the best performance.

In supervised settings with the same LM configuration, 300 words of user transcription is sufficient to outperform unsupervised adaptation. By selecting the utterances with the lowest confidence for transcription, we can further reduce the effective transcription WER by another 0.6%.

For future work, we plan to present the efficient lattice rescoring data structure and algorithms that enable us to practically conduct the above experiments. We also would like to explore using both the user transcriptions and recognition hypotheses for LM adaptation. In addition to using the hypotheses for text selection, we hope to examine the use of word-level confidence scores as n-gram weighting features to train a component model from the hypotheses.

Chapter 7

Data Structure & Algorithms

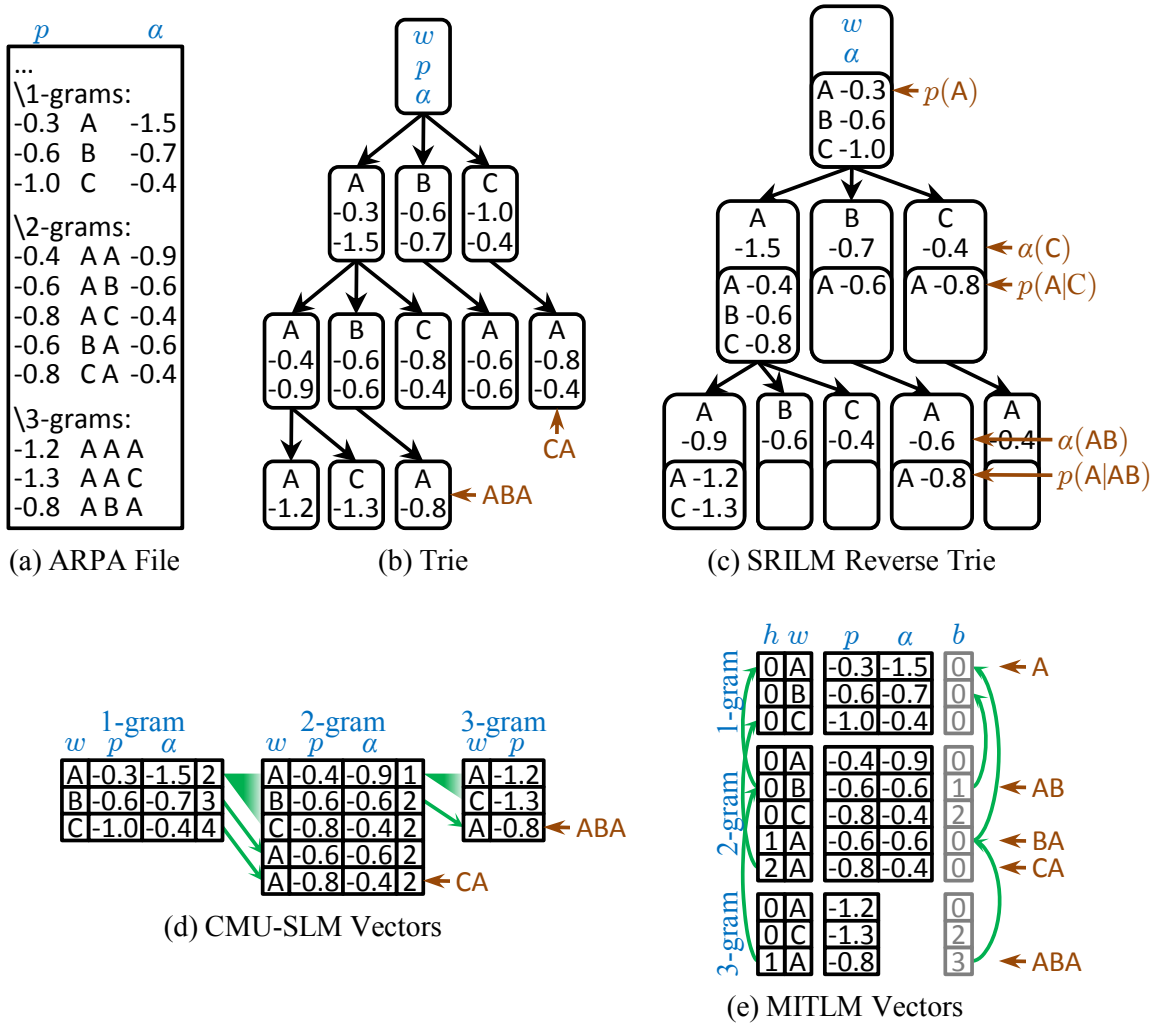
7.1 Motivation

In the previous chapters, we introduced the generalized linear interpolation (Chapter 4) and n-gram weighting (Chapter 5) techniques to improve language model estimation in domains with limited matched training data. However, both approaches introduce additional model parameters that need to be tuned on an in-domain development set. While unsupervised and active learning techniques (Chapter 6) can be applied to obtain development data, an efficient implementation is required to tune the model parameters to optimize the development set performance. Otherwise, the proposed techniques will achieve little, if any, practical significance.

In fact, many of the most effective existing techniques for n-gram language modeling, such as modified Kneser-Ney smoothing [22] and count merging interpolation [1], also benefit from efficient parameter tuning. Oftentimes, generic numerical optimization techniques [123] can be applied to iteratively tune the model parameters to optimize the development set perplexity. However, due to the lack of support for performing such optimization in popular language modeling toolkits, such as the SRI Language Modeling (SRILM) [138] toolkit, most existing research in this field opts for simpler techniques with inferior results.

In this chapter, we propose a data structure designed specifically for language model (LM) training algorithms involving iterative parameter tuning. By taking advantage of the iterative nature of such optimization, we demonstrate how to apply the techniques of result caching, sufficient statistics, and computation masks to reduce the amount of computation. Using the resulting MIT Language Modeling (MITLM) toolkit, we not only significantly outperform the performance of SRILM, but also enable more sophisticated language modeling algorithms previously considered impractical.

In the following sections, we first compare traditional n-gram data structures with our proposed representation. Next, we present efficient implementations of various LM algorithms, including modified Kneser-Ney smoothing, linear interpolation, and perplexity evaluation. After comparing the performance of MITLM with SRILM on a set of basic LM tasks, we demonstrate the effectiveness and practicality of tuning more sophisticated LMs. Finally, we end with ideas for future optimization and toolkit improvement.



(a) ARPA File (b) Trie (c) SRILM Reverse Trie

(d) CMU-SLM Vectors

(e) MITLM Vectors

Figure 7-1: Various n-gram data structures. w: word, p: probability, α: backoff weight, h: history index, b: backoff index.

7.2 Data Structure

An n-gram LM represents the probability of a word sequence $w_1^N = w_1, \ldots, w_N$ as $p(w_1^N) = \prod_{i=1}^{N} p(w_i|w_1^{i-1}) \approx \prod_{i=1}^{N} p(w_i|h_i)$, where n is the model order and $h_i = w_{i-n+1}^{i-1}$ represents the history for word w_i. Since many n-grams are not observed in the training data, we can smooth the maximum likelihood estimate by distributing probabilities from seen to unseen n-grams and assigning probabilities proportional to the lower-order model to the unseen n-grams. For these cases, $p(w|h) = \alpha(h)p(w|h')$, where $\alpha(h)$ is the backoff weight and h' is the backoff history obtained by removing the earliest word from h. The resulting backoff n-gram LM is commonly represented in the ARPA text file format [139], which stores $p(w|h)$ and $\alpha(h)$ of the observed n-grams and their histories, as shown in Figure 7-1a. The probabilities and backoff weights are stored in $\log_{10}$. Backoff weights for unseen n-gram histories are assigned the value $1 = \log_{10} 0$. Thus, in this example, we can compute the following probabilities:

$$
\begin{aligned}
p(A|AA) &= 10^{-1.2} \\
p(C|BA) &= \alpha(BA)p(C|A) = 10^{-0.6-0.8} \\
p(A|CB) &= \alpha(CB)p(A|B) = 10^{0-0.6} \\
p(C|AB) &= \alpha(AB)p(C|B) = \alpha(AB)\alpha(B)p(C) = 10^{-0.6-0.7-1.0}
\end{aligned}
$$

7.2.1 Existing Representations

Previous work on data structures for n-gram models has primarily focused on runtime performance and storage compression [24, 140, 150]. With the availability of the Google Web 1T 5-gram corpus [10], recent research has begun to examine efficient and distributed representations for building large-scale language models [41, 42, 114]. However, so far, these efforts only support simple, but subpar, smoothing and interpolation techniques, few that involve iterative parameter optimization.

While the ARPA file format serves as a standard cross-implementation representation for backoff n-gram language models, it is inefficient as a runtime data structure. A simple option is to represent the model as a trie, or prefix tree, where each n-gram node stores the word index w, conditional probability $p(w|h)$, and backoff weight $\alpha(hw)$ (Figure 7-1b). To reduce the number of lookups needed to compute n-gram probabilities involving backoff, SRILM maintains the n-gram histories backwards within the trie and stores the conditional probabilities in a dictionary within each n-gram history node (Figure 7-1c). Although simple to build incrementally, tries demonstrate poor memory locality during node traversals and cannot be serialized efficiently. Instead, the CMU-Cambridge Statistical Language Modeling (CMU-SLM) toolkit [26] utilizes a compact vector encoding of the n-gram trie [150] where pointers to the child nodes are encoded with array indices (Figure 7-1d). The resulting structure not only improves memory locality, but also reduces the memory footprint.

7.2.2 Ideal Properties

Existing data structures have been primarily designed for efficient n-gram probability evaluations at runtime. However, iterative LM estimation algorithms repeatedly traverse through the n-grams at each level, using n-gram features such as counts to compute intermediate values, conditional probabilities, and backoff weights. Thus, an ideal data structure for LM estimation requires simple and fast iterators and allows intermediate values to be associated

with each n-gram. Specifically, given that many computations involve statistics from the corresponding history and backoff n-grams, links to these n-grams need to be pre-computed for each n-gram to avoid duplicate lookups.

Furthermore, an ideal data structure needs to minimize memory footprint while supporting efficient file serialization and memory-mapped model loading. To optimize the use of hardware resources, it should preserve memory locality for common algorithms, utilize vector instructions, and support multi-core architectures. Finally, to enable the estimation of LMs from extremely large corpora, both the data structure and algorithms have to support parallelization over distributed memory computer clusters.

7.2.3 MITLM Vectors

In this work, we present a novel data structure designed specifically for LM estimation procedures with iterative parameter tuning. Instead of dynamically building a trie in memory like SRILM, the proposed data structure represents n-grams as a list of vectors, one for each order. Unlike CMU-SLM, each element of the vector represents a specific n-gram by storing the target word index and the index of the history n-gram (Figure 7-1e). Conceptually, we can view this as a compact vector encoding of an n-gram trie with reversed pointers such that each node points to its parent. Unlike standard tries, we can efficiently determine the words of any n-gram node by following the history indices without costly searches.

With a vector representation, traversing over the n-gram structure reduces to index increments and preserves memory locality. Data such as count and backoff n-gram index can be easily associated with each n-gram by storing them in corresponding elements of separate data vectors. Thus, computations like backoff weights that involve statistics from the history and backoff n-grams can access these values via simple index lookups without costly searches. By using indices to link n-grams to their history and backoff n-grams, the resulting vector data structure can be trivially serialized to a binary data file representation and even loaded via memory mapping.

To construct the n-gram vector data structure from a list of n-grams encountered while processing a text corpus or reading a counts file, we need a dictionary data structure that can efficiently map an n-gram, represented by its history n-gram and target word indices, to its associated index within the vector representation. Thus, when inserting an n-gram $w_1w_2w_3$ into the data structure, we iteratively lookup the unigram index i_1 from $(0, w_1)$, bigram index i_2 from (i_1, w_2), and trigram index i_3 from (i_2, w_3). If any of these n-grams is not found, we add it to the end of the vector and return the corresponding index.

While we can use a hash table to map the history/word index pair key to the n-gram index value, traditional implementations require storing a copy of the key. In our representation, since the key can be recovered from the index value via simple lookup in the n-gram vector, we only need to maintain an array of index values that stores the index of the n-gram at the position specified by the hash value of the history/word index pair key, significantly reducing the space used by the hash table. To address collisions in the hash value, we apply quadratic probing [27], but can also consider cuckoo hashing [117].

With the compact hash vector data structure, we can efficiently construct the n-gram vectors and pre-compute the backoff n-gram indices. Since most LM algorithms actually do not require costly lexical lookups during the iterative computation given direct access to the history and backoff n-grams, such pre-computation significantly improves performance. Furthermore, as both the hash vector backoff n-gram indices can be reconstructed from the history/word vectors, they do not require additional storage during serialization.

Operations	Examples / Description
$+, -, \times, \div, \cdot, \log, \exp, \sum$	$x = [1, 2, 3]; \quad y = [4, 5, 6]; \quad x + y \Rightarrow [5, 7, 9]; \quad \sum x \Rightarrow 6$
Indexing	$x[0, 2] \Rightarrow [1, 3]; \quad x[x > 2] \Rightarrow [3]$
$y = \text{bincount}(x)$	$y = \text{binweight}(x, 1)$
$y = \text{binweight}(x, w)$	for i in range(len(x)): $y[x[i]]$ += $w[i]$
$y = \text{binlookup}(x, t, d{=}0)$	for i in range(len(x)): $y[i] = (0 \leq x[i] < \text{len}(t))\ ?\ t[x[i]] : d$
$y = \text{binset}(x, m)$	for i in range(len(x)): if $m[i]$: $y[x[i]] = 1$

Figure 7-2: Common vector operations in LM estimation.

7.3 Algorithms

Once the data structure has been constructed, most LM estimation algorithms can be decomposed into simple vector operations (Figure 7-2) on the n-gram counts and history/backoff indices without referring to the word indices. For algorithms involving iterative estimation of model parameters, intermediate values can often be cached across iterations to reduce computation. As the model parameters are tuned to minimize an objective function such as the development set perplexity, we can further improve the performance by masking the operations to only the subset of n-gram values that affect the result during the iterative optimization. Lastly, in iterative procedures, it may be worthwhile to first compute sufficient statistics that summarize the full data to reduce the work involved per iteration. In the following sections, we will describe example applications of the above techniques to a few common LM estimation approaches.

7.3.1 Modified Kneser-Ney Smoothing

Modified Kneser-Ney smoothing [22] is one of the best performing techniques for n-gram LM estimation. Unlike the original Kneser-Ney smoothing [90] where a constant is subtracted from each count, modified Kneser-Ney subtracts a different value depending on the actual count. Specifically, it assigns probability $p(w|h) = \frac{\tilde{c}(hw) - D(\tilde{c}(hw))}{\sum_w \tilde{c}(hw)} + \alpha(h)p(w|h')$, where $\tilde{c}(hw)$ is the modified Kneser-Ney n-gram count, $D(\cdot)$ is the discount function, and $\alpha(h)$ is the backoff weight satisfying $\sum_w p(w|h) = 1$ (see Section 2.1.3).

As shown in Figure 7-3, we can compute all probabilities and backoff weights simultaneously using efficient vector operations. For example, the Kneser-Ney counts $\tilde{c}(hw)$, defined by the number of unique n-grams with hw as their backoffs, can be computed using a bincount operation on the higher order backoff indices b^+. Each higher order n-gram contributes a count of 1 to its backoff n-gram, as desired. With aligned vectors, we can also compute the backoff probabilities for the current order by simply indexing the lower order probability vector p^- by the backoff indices b and multiplying it by the backoff weights.

Instead of estimating discount factors $f[i] = D(i)$ from count statistics, we can also tune them to minimize the development set perplexity, which has been observed to improve performance [22]. In each iteration of the parameter optimization, it is sufficient to repeat only steps 5–7, as the inverse history counts $c_h(h) = \frac{1}{\sum_w \tilde{c}(hw)}$ are independent of the parameters f. Thus, by caching c_h in the n-gram vector structure, we significantly reduce the amount of computation within each iteration.

1.	$\tilde{c} = \mathit{highestOrder}\ ?\ c : \text{bincount}(b^+)$	# Load/compute counts
2.	$n = \text{bincount}(\tilde{c}); Y = \frac{n[1]}{n[1]+2n[2]}$	# Gather count statistics
3.	for i in (1,2,3): $f[i] = i - (i+1)Y\frac{n[i+1]}{n[i]}$	# Estimate default discount params
4.	$c_h = 1/\text{binweight}(h, \tilde{c})$	# Compute inv history counts
5.	$d = \text{binlookup}(\tilde{c}, f, f[3])$	# Lookup discounts
6.	$\alpha = \text{bincount}(h, d) \times c_h$	# Compute backoff weights
7.	$p = (\tilde{c} - d) \times c_h[h] + \alpha[h] \times p^-[b]$	# Compute interpolated probs

Figure 7-3: Modified Kneser-Ney smoothing. b: backoff indices, b^+: higher order backoff indices, h: history indices, c: n-gram counts, $\tilde{c}$: modified counts, c_h: inverse history counts, n: count of counts, f: discount factors, d: n-gram discounts, α: backoff weights, p: probabilities, p^-: lower order probabilities.

1.	for i in $[1,2]$: $p_i, \alpha_i = \text{loadlm}(lm_i, \mathit{model})$	# Load n-gram models
2.	for i in $[1,2]$: $z = (p_i == 0); p_i[z] = \alpha_i[h[z]] \times p_i^-[b[z]]$	# Compute missing probs
3.	$p = p_1 \times w_1 + p_2 \times w_2$	# Interpolate probs
4.	$\alpha = (1 - \text{binweight}(h, p))/(1 - \text{binweight}(h, p^-[b]))$	# Compute backoff weights

Figure 7-4: Linear interpolation. LMs are loaded into a common n-gram structure such that they share common history and backoff indices, h and b. p_i, p_i^-, α_i: probabilities, lower order probabilities, and backoff weights of i^{th} LM; w_1, w_2: interpolation weights; p, α: interpolated probabilities and backoff weights.

7.3.2 Linear Interpolation

Linear interpolation [81] is the most popular technique for merging multiple n-gram LMs. To create a static backoff n-gram LM from component LMs, SRILM interpolates the component probabilities for the union of the observed n-grams, using backoff probabilities if necessary, and computes the backoff weights to normalize the model (see Section 2.3.2 for more details). Figure 7-4 contains an efficient vector implementation of the linear interpolation algorithm.

The interpolation weights are typically chosen to minimize the development set perplexity using an iterative algorithm, such as Expectation-Maximization (EM) [33] or numerical optimization techniques [123]. Since computing the development set perplexity typically only involves a small subset of n-gram probabilities and backoff weights, we can pre-compute Boolean masks m_p and m_α to represent this subset and limit the computation to only those elements specified by the masks. Because the backoff weight computation (step 4) involves other n-gram probabilities, we need to extend the probability mask to include these n-grams. Specifically, computing $\text{binweight}(h, p)$ with mask m_α requires the probabilities of n-grams whose histories are in m_α. Thus, we need to extend the mask via $m_h = \text{binlookup}(h, m_\alpha)$, m_p |= m_h. Likewise, to compute $\text{binweight}(h, p^-[b])$, we shall extend the mask of the lower order probability vector using m_{p^-} |= $\text{binset}(b, m_h)$. As the development set generally only contains a small subset of the overall n-grams, we can now substantially reduce the computation per iteration in steps 3 and 4 by limiting it to only the masked n-grams. Similar techniques can be applied to smoothing and other algorithms.

1.	$c_p, c_\alpha, N, N_{oov} = \text{loadevalcorpus}(corpus, lm)$	# Pre-compute count stats
2.	$p, \alpha = \text{estimatelm}(lm, params)$	# Estimate LM
3.	$perplexity = \exp[-\frac{1}{N}(c_p \cdot \log p + c_\alpha \cdot \log \alpha)]$	# Compute perplexity

Figure 7-5: Perplexity evaluation. Sufficient statistics c_p and c_α are computed as the contribution each LM probability and backoff weight makes towards the corpus perplexity. N and N_{oov} are the total counts of in-vocabulary and out-of-vocabulary words in the corpus, respectively.

7.3.3 Perplexity Evaluation

As described in Section 2.1.2, the perplexity of a LM evaluated on a corpus w_1^N can be computed as:

$$PP = \exp\left[-\frac{1}{N}\log\prod_{i=1}^{N} p(w_i|h_i)\right]$$

Given that not all n-grams in the corpus may be observed in the LM, we may need to rely on backoff probabilities to compute $p(w_i|h_i)$.

Finding an n-gram from word indices is a costly procedure. Instead of performing this lookup in every iteration of perplexity optimization, we can pre-compute sufficient statistics that describe the evaluation corpus, as they remain constant across iterations. Specifically, since the corpus probability can be decomposed into the product of observed $p(w|h)$'s and $\alpha(h)$'s, we can represent the corpus compactly using the count each appears in the product, or $c_{p(w|h)}$ and $c_{\alpha(h)}$. Thus, after updating the LM probabilities and backoff weights in each iteration, we can evaluate the LM perplexity efficiently using vector dot products, as shown in step 3 of Figure 7-5. The counts can also serve as the n-gram masks for efficient LM estimation.

7.4 Implementation

Using the compact data structure and optimized algorithms described in the previous sections, we have implemented the MITLM toolkit, available as open source software at `http://www.sls.csail.mit.edu/mitlm/`. Currently, it supports various iterative LM estimation algorithms, including modified Kneser-Ney smoothing [22], linear interpolation [81], count merging [1], generalized linear interpolation (Chapter 4), and n-gram weighting (Chapter 5). Model parameters can be tuned to minimize the development set perplexity using numerical optimization techniques such as Powell's method [123] or L-BFGS-B [159]. Alternatively, we can tune the parameters directly on the word error rate, efficiently computed via optimized lattice rescoring. Finally, to reduce the time spent reading and writing LMs, the toolkit supports fast binary serialization and optional file compression.

For efficiency, the core data structures and I/O operations are implemented in C++. To evaluate vector expressions efficiently without allocating temporary storage, we designed a vector library that utilizes expression templates to perform element-wise evaluation [98,146]. In addition to standard vector operations, the library also supports conditional and masked evaluation. Although the currently supported algorithms have been ported to C++ for efficiency, in the future, we plan to provide a Python wrapper that allows researchers to experiment with additional algorithms interactively using a high-level programming language. Currently, the toolkit does not support any parallelism.

Dataset	Words	1-grams	2-grams	3-grams
BNDev	3,573,908	56,156	778,556	1,973,886
BNTest	13,931,084	110,079	2,028,988	6,158,630
BNTrain	131,668,976	355,995	9,153,440	37,884,316
NYT96	156,879,556	319,279	10,939,278	38,566,120

Table 7.1: Summary of evaluation datasets.

	Load Time (sec)		Save Time (sec)		File Size (MB)	
BNTrain	SRILM	MITLM	SRILM	MITLM	SRILM	MITLM
Text Corpus	71	142	–	–	725	725
ARPA LM	47	79	74	37	1434	1420
ARPA LM (gzip)	50	85	71	34	423	438
Binary LM	15	10	37	1.7	397	838
+ Hash/Backoff		1.5		3.7		1710

Table 7.2: File I/O performance of SRILM and MITLM.

7.5 Experiments

To demonstrate the capabilities of the MITLM toolkit, we will evaluate the runtime performance of select file I/O, smoothing, and interpolation experiments using a 2.66 GHz CPU on the broadcast news domain. Specifically, the target corpus consists of the Broadcast News corpus [63] from 1996, where we designated the first month as the development set (BNDev) and used the remaining five months as the test set (BNTest). For training data, we use the Broadcast News data from 1992 to 1995 (BNTrain), along with the New York Times articles [64] from 1996 (NYT96). Table 7.1 summarizes all the evaluation data.

7.5.1 File I/O

In Table 7.2, we compare the performance of loading and saving the BNTrain trigram model in various formats using SRILM and MITLM. Due to the difference in data structure, MITLM is slower than SRILM when constructing an n-gram model from a text corpus or an ARPA LM file. However, with more efficient iterators (and optimized file output functions), MITLM makes up for the difference in faster writes to an ARPA LM file. On multi-core processors, the use of gzip compression reduces the file size by more than 3x with negligible impact on the overall time. The small difference in the ARPA LM file sizes between SRILM and MITLM is due to the occasional difference in the precision of the floating point text representations.

Instead of converting the internal LM data structure to an ARPA text file and later reconstructing the internal representation via costly text parsing, we can achieve significant speedups by saving the data structure to a binary file instead. Using such a representation, SRILM reduces the I/O time by about a factor of 3. By directly serializing the vector data structure to a binary file and recomputing the hash table and backoff n-gram indices at load time, MITLM further reduces the load and save time by 30% and 20x, respectively. If we instead store the hash table and backoff n-gram indices, we can further improve the load time by 10x over SRILM. Using a simple vector structure, MITLM significantly outperforms SRILM in terms of binary I/O performance.

7.5.2 Smoothing

In [22], Chen and Goodman conclusively showed that modified Kneser-Ney smoothing achieves better performance with tuned discount parameters, especially in the presence of mismatch between the training and test data. However, perhaps because popular language modeling toolkits, such as SRILM, do not directly support parameter tuning, most subsequent publications using this smoothing algorithm estimated the discount parameters from count statistics instead.

Estimating a trigram LM using modified Kneser-Ney smoothing with fixed parameters on the BNTrain corpus takes about 4.6 minutes using SRILM, with a peak memory usage of 3.2 GB. With MITLM, the performance improves considerably to 3.0 minutes and 1.9 GB. Excluding time spent in file I/O, the difference is even more dramatic with SRILM and MITLM spending 127 and 3.1 seconds, respectively. By using a vector representation with pre-computed backoff indices, MITLM improves the efficiency of LM smoothing by more than an order of magnitude.

Instead of using fixed values, we can also tune the 9 modified Kneser-Ney discount parameters to minimize the development set perplexity using Powell's method. For this dataset, applying masking to compute only those n-grams needed for perplexity evaluation reduces the vector computation by up to a factor of 26x, enabling the 240 iterations of the parameter optimization process to complete in only 2.9 minutes and 2.2 GB of memory.

With its mostly sequential memory accesses, MITLM also improves memory locality over SRILM. For example, although both toolkits significantly exceed the available 8GB of memory when estimating a trigram model on the full 1.1 billion words of New York Times articles from 1994 to 2004 [64], MITLM manages to finish in 30 minutes, whereas SRILM took over 24 hours due to disk thrashing. By designing the data structure for iterative parameter optimization, we can significantly improve the runtime performance and enable previously impractical parameter tuning algorithms.

7.5.3 Interpolation

SRILM supports building a static backoff LM from the linear interpolation (LI) of component LMs. It also provides a separate script for tuning the interpolation weights to minimize the development set perplexity using Expectation-Maximization. Using SRILM, we are able to tune and interpolate BNTrain and NYT96 into a single LM in 19.5 minutes and 3.6 GB via 18 iterations of the EM algorithm. By integrating the tuning process into the estimation procedure, MITLM interpolates the same LMs via 10 iterations of L-BFGS-B in 8.3 minutes and 5.4 GB, spending only 1.8 minutes on parameter optimization. Since SRILM optimizes the parameters for dynamic interpolation, it uses less memory than MITLM, but generally yields suboptimal results for static interpolation.

Extending beyond the capabilities of current publicly available language modeling toolkits, Table 7.3 lists the performance and computation times of tuning more sophisticated LM interpolation models. By adjusting the interpolation weight as a function of features derived from the n-gram history, count merging (CM) [1] and generalized linear interpolation (GLI) (Chapter 4) improve the test set performance with only a moderate increase in computational resources. In fact, both interpolation techniques using MITLM complete in less time than linear interpolation using SRILM.

Technique	Time (min)	Memory (GB)	Perplexity	# Parameters
LI	8.3	5.4	125.8	1
CM	9.8	7.2	125.5	1
GLI: $\log_{1+}(\sum_w c(hw))$	16.8	7.4	124.7	3

Table 7.3: Performance of MITLM for various interpolation techniques.

7.6 Summary

By designing the n-gram data structure and algorithms for iterative LM estimation, we have implemented an efficient language modeling toolkit that supports parameter optimization for modified Kneser-Ney smoothing and various LM interpolation techniques. Compared with the popular SRILM toolkit, the resulting MITLM toolkit improves the running time of the modified Kneser-Ney algorithm by 40x and reduces the memory usage by 40%. Furthermore, it enables the efficient optimization of parameters for more sophisticated interpolation techniques, instead of relying on empirical trial and error approaches [1, 48].

Although MITLM is not designed to provide efficient runtime lookup, the ARPA LM output can be easily converted into other formats more suited for such applications. As the toolkit is relatively immature, it currently lacks many of the essential features for building practical language models, such as vocabulary filtering and n-gram pruning. Thus, for future work, we would like to develop efficient implementations of these algorithms. To enable researchers to more easily experiment with language models, we plan to provide a Python wrapper for interactive evaluation and computation. Finally, we hope to further optimize the memory usage and parallelize the implementation to support large-scale LM estimation.

With the simple vector representation of n-grams, we invite researchers to implement future language modeling techniques using MITLM. We hope that with the public release of this toolkit, future research will also consider more effective language model estimation techniques that are previously impractical.

Availability The latest version of the MIT Language Modeling toolkit and accompanying documentation are available as open source software at `http://www.sls.csail.mit.edu/mitlm/`.

Chapter 8

Conclusion & Future Work

8.1 Contributions

In this thesis, we studied the problem of language modeling in domains with limited matched data, such as lecture transcription. We observed that despite the lack of a matched training corpus, data that partially matches the target domain in topic or style is often available. To take advantage of the partially matched data, we proposed various language modeling techniques that mitigate the mismatch between the training data and the target domain.

Without matched data, most language modeling techniques involve combining models trained from partially matched corpora. As the training components generally match the target domain in different ways, they are not equally predictive across all n-gram contexts. Thus, instead of using a set of fixed interpolation weights across all contexts, we proposed the *generalized linear interpolation* technique (Chapter 4) that models the interpolation weights as a function of n-gram context features. Using count and topic features, we succeeded in reducing the test set WER by up to 0.8% and 0.3% over simple linear interpolation and count merging, respectively.

As the data from which the component language models are trained is partially mismatched, not all the n-grams observed from these corpora are equally relevant to the target domain. Thus, we introduced the *n-gram weighting* technique (Chapter 5) that weights each observed n-gram by a relevance factor estimated as a function of various n-gram features. Using the distribution features derived from the segmentation and metadata information readily available with the training corpora, we further improved the test set WER by up to 1.1% and 0.7% over simple and generalized linear interpolation, respectively. Combining Textbook, Lectures, and Switchboard, generalized linear interpolation with n-gram weighting reduces the WER from 33.3% with simple linear interpolation to our best result of 31.9%.

The above improvements assume the existence of a large in-domain development set for parameter optimization. In situations where no in-domain data is available (Chapter 6), we can perform multiple recognition passes and tune the model parameters using the 1-best recognition hypotheses from the previous pass. This unsupervised technique is able to achieve within 0.2% of the oracle WERs, obtained by tuning the parameters on the target lectures, within four recognition iterations. Alternatively, in certain application scenarios, we may be able to ask the users to transcribe utterances from the target lectures for use as development data. In our experiments, 300 words of user transcription are sufficient to outperform unsupervised adaptation. By transcribing the least confident utterances first, we further reduce the effective WER by 0.6%.

Sophisticated language modeling techniques, such as the ones presented in this thesis, are only useful if they can be efficiently implemented and practically applied without exorbitant computational resources and significant human intervention. Thus, to improve the performance of various LM estimation procedures involving iterative parameter tuning, we designed an efficient vector-based n-gram data structure and outlined a set of optimization techniques (Chapter 7). With the resulting MIT Language Modeling (MITLM) toolkit, we improved the running time of the modified Kneser-Ney smoothing algorithm by 40x and reduced the memory utilization by 40%. Furthermore, we enabled the efficient optimization of parameters for more sophisticated techniques previously deemed impractical, as exemplified by the following description of the count merging weights:

> *The n-gram counts can be empirically weighted to minimize the perplexity on a set of development data . . . it has to be done by trial and error and cannot easily be optimized*

The MIT Language Modeling toolkit is currently available as open source software at `http://www.sls.csail.mit.edu/mitlm/`. It supports the following:

- Smoothing – Maximum likelihood, Kneser-Ney family
- Interpolation – Linear, count merging, generalized linear interpolation
- n-gram weighting
- Evaluation – Perplexity, word error rate
- Optimization – Powell's method, L-BFGS, L-BFGS-B
- Lattice rescoring
- Binary Data Formats – Counts, ARPA LM, lattices

8.2 Future Work

8.2.1 Discriminative Training

In the experiments presented in this thesis, we often observed that the model that minimizes perplexity does not always yield the lowest word error rate. Thus, for future work, we would like to explore discriminative approaches for tuning language model parameters. Using lattice rescoring, we can efficiently compute the word error rate of the recognition lattices for the development set as we iteratively adjust the language model parameters. Since WER is a piecewise constant function of the language model parameters, gradient based numerical optimization schemes generally fail. However, bracketing techniques such as Powell's method may be effective, especially when initialized with the parameter values optimized on the development set perplexity.

Instead of directly optimizing on WER, recent research in discriminative training for acoustic models suggest that better generalization may be obtained by tuning model parameters to a combination of empirical WER and the discriminative margin [100, 156]. As the large-margin objective is piecewise continuously differentiable, gradient-based numerical optimization techniques can now be employed for more efficient parameter tuning. Preliminary results suggest that an additional 0.1% WER reduction may be obtained for generalized linear interpolation using discriminative parameter tuning.

8.2.2 Parallelized Implementation

In building the MIT Language Modeling toolkit, we designed a compact vector-based n-gram representation and decomposed common language modeling algorithms to a set of basic vector operations. This not only enables the efficient estimation of language models that require iterative parameter optimization, but also allows the computation to be more easily parallelized via CPU vector instruction, parallel multi-core processing, and even distributed computation across computer clusters.

As the computational infrastructure and available training data increase in scale, there is growing interest in building ever larger language models, especially for machine translation. Recent research has begun to examine efficient and distributed representations for building large-scale language models [41, 42, 114]. However, so far, these efforts only support simple, but subpar, smoothing and interpolation techniques, few that involve iterative parameter optimization. Thus, for future work, we would like to modify MITLM to support parallelization across distributed-memory computer clusters. Specifically, with an appropriate vector library abstraction, we hope to hide all the implementation complexities within the distributed vector library implementation and reuse the same high-level vector implementation of common language modeling algorithms.

8.2.3 Hierarchical Modeling

In the description of the models proposed in this thesis, we did not apply any regularization to constrain the model parameters and avoid overfitting. Furthermore, we did not take advantage of the n-gram backoff structure to improve the robustness of the estimated parameters and the resulting model. Intuitively, for both generalized linear interpolation and n-gram weighting, the interpolation weights $\lambda(h)$ and weighting factors $\beta(hw)$ should both be biased by the corresponding lower-order backoff values $\lambda(h')$ and $\beta(h'w)$ as priors. Thus, we would like to evaluate the effectiveness of applying hierarchical modeling techniques, such as hierarchical logistic regression [153] and hierarchical maximum entropy [36], to generalized linear interpolation and n-gram weighting in future work.

8.3 Applications & Extensions

Although this thesis primarily focuses on language modeling for lecture transcription, the techniques presented here can also be applied to other domains with limited data. For example, in voice search, although transcripts of voice search queries may be limited, we generally have access to a large quantity of web search query logs and perhaps closed caption transcripts from TV quiz shows. By efficiently combining data from partially matched training corpora using generalized linear interpolation and n-gram weighting, we can improve the accuracy of the resulting voice search application.

In addition to language modeling, the intuitions from the proposed techniques can also be transfered to other research areas, such as acoustic modeling for speech recognition and semantic decoding in natural language understanding. Similar to an n-gram model, an acoustic model is composed of multiple component distributions, one per phone/context pair. Analogous to language modeling, acoustic model training data that matches both the speaker and acoustic environment of the target condition is rarely available in sufficient quantity. Thus, we face the same challenge of how to best utilize partially matched data in the estimation of the acoustic model.

When interpolating language models, we observed that the interpolation weight depends on the observation count of each component in a non-linear way. Thus, when combining acoustic models trained from partially matched conditions, the interpolation weight should increase with the number of observations. However, with sufficiently many observations to estimate the model, the component relevance should dominate the interpolation weight.

Just as some n-grams better characterize the target domain than others, not all acoustic observations from partially matched data equally reflect the target distribution. For example, the defining features of vowels may be more speaker-dependent than consonants. Thus, when combining data from both matched and mismatched speakers, we may want to de-emphasize the weight placed on the vowels from the mismatched speakers.

8.4 Vision

Building successful spoken dialog systems requires effective modeling of the underlying user processes. While these processes often show complex variations across a population, requiring significant amounts of data to model properly, individuals often exhibit consistencies that can be learned with modest amount of correction feedback. In many applications, such feedback can be obtained with minimal effort and technical knowledge from the user. Combining data collected from individual users within a large community further enables the construction of more complex models that better describe these user processes. A collaborative adaptation framework not only distributes the effort required to train the system across the user community, but also improves the individual user models and the shared base models via online adaptation.

We started this thesis with a vision of diverse and widespread spoken dialog applications, personalized to individual users via rapid adaptation. However, the lack of sufficient matched training data limits the effectiveness of traditional modeling techniques. Thus, as a first step towards the grand vision, we examined in this thesis various language modeling techniques to improve the speech recognition accuracy in domains with limited data. Yet, despite the progress we have made, many challenges remain. Below, we summarize some of the key research problems we have identified.

8.4.1 Personalized Semantic Decoding

Once the user utterance has been recognized, a spoken dialog system needs to process the input sentence and perform an appropriate action based on the decoded semantics. While extensive research has investigated the extraction of semantic content from spoken language (see [149] for a review), most existing approaches require significant amount of training data with fully annotated parse trees to build effective models. Given the limited availability of training data in our envisioned scenarios, we need techniques that better generalize from single examples and take advantage of partially matched data.

Recently, Ye and Young proposed a clustering-based approach that clusters training sentences into a set of prototype templates and extracts concepts from new sentences by matching them against the prototypes using dynamic time warping [154]. Although simpler than many other statistical models, preliminary results indicate significantly better concept extraction performance. Although not explored in the original paper, we believe that this technique has more potential to scale down to scenarios with limited training data and take advantage of partially matched data using intuitions introduced in this thesis.

8.4.2 Concept Induction

In addition to labeled training examples, semantic decoding also requires a list of concepts and possible values. While we may be able to hard-code certain concepts (date, time, numbers) and pre-define other frequently used concepts (cities, contacts, stocks), it is impractical for system developers to enumerate all possible concepts users are likely to specify. Thus, in order to achieve the envisioned application diversity, the system needs to be able to automatically generate concepts from user feedback. Specifically, given a few examples of a concept, how can we hypothesize other members of the concept?

This problem of *set expansion* is perhaps best exemplified by the web application *Google Sets* [61]. Given *Boston* and *San Francisco* as example elements in the set, it is able to generate *Chicago*, *New York*, and other U.S. cities as additional members. More recent research by Wang and Cohen has also leveraged the structure of web pages to better identify set membership [148]. Existing work in set expansion tends to evaluate the precision of the resulting set given fixed seed examples. However, in an interactive application, the system may be able to propose related instances and ask the user to confirm their membership within the set. In this active learning scenario, a more relevant metric may be the minimum number of examples required before the system can sufficiently identify the set.

8.4.3 Collaborative Online Adaptation

When building models for a spoken dialog system, we have data from not only the target user, but also a community of users using the same system. Traditional approaches in this situation tend to adapt a background model trained from all data with the data specific to the target user. While it can be effective, this approach does not scale well when we want to model all users simultaneously in an online fashion for rapid adaptation.

Simultaneously estimating multiple personalized models while building a global model belongs to the recently introduced class of machine learning problems coined *multi-domain learning* [35]. In that work, Dredze and Crammer developed an online algorithm for jointly learning multiple related linear classifiers. Although not directly applicable, similar intuitions can be applied to language modeling, semantic decoding, and other natural language processing tasks with great promise. By learning from the data across all users, we can build more effective models with fewer training instances from each target user. By effectively adapting all user models with each new observation, we hope to convince users that it is worth their effort to provide feedback and train the system.

8.5 Final Thoughts

We envision a future where users can interact with machines via a spoken natural language interface. Instead of relying on software developers with spoken dialogue design experience to develop expert systems in a few select domains, the system learns from the feedback provided by the community of users and personalizes its behavior to the individual preferences of each user. With interfaces that enable users to provide correction feedback and adaptation algorithms that immediately learn from such feedback, we hope to break the chicken and egg deadlocks over data collection and application development. With diverse spoken dialogue applications that focus on real user needs, we look forward to a future when users view spoken dialogue applications as essential productivity tools, not just high-tech toys.

Appendix A

Collaborative Data Collection

Community participation is a key distinctive feature of Web 2.0, where the users contribute a significant value to the overall web site. In addition to the social aspects enabled by a diverse community of users, positive network effects also can result from the content contributed by such a community. In fact, Wikipedia, Flickr and MySpace are all prime examples of Web 2.0 applications built on such content. Similarly, the entire open source movement is based on the concept of collaborative development where participants contribute code to the development community for the benefit of everyone. While it is clear that these community-based services can generate significant values for their users, it is less clear how companies can generate revenue from this model, beyond paid advertisements and support services. In this section, we will explore a specific model of community participation, where the user contribution is in the form of data for building data-driven applications rather than for direct end-user consumption. Specifically, we will consider a hypothetical GPS navigation service based on community participation and examine potential business models for such a service.

A.1 Data-Driven Applications

With the recent advances in distributed algorithms and statistical learning, more and more commercial software are starting to employ artificial intelligence and machine learning techniques to reduce human involvement and improve system performance. For example, recommender systems apply a machine learning technique known as collaborative filtering to predict individual user ratings and recommend new items to users based on previous item ratings from a community of users. Of all the commercial recommender systems, Amazon's product recommendation feature is perhaps the most well-known. By identifying products of interest to other users with a similar profile, the system is able to automatically provide personalized product recommendations based on the products the user has previously clicked on or rated. As with most machine learning algorithms, significant amount of training data is required to achieve desirable performance. Thus, for many data-driven applications, data collection represents a key challenge to successful commercial deployment.

Traditionally, data collection is often viewed as a labor intensive and expensive process. However, with the advent of the Web 2.0 culture, users are increasingly willing to contribute information to improve the overall value of the system. Furthermore, not all data collection involves explicit feedback from the user. For example, recommender systems can collect data on the user interactions with the system (search queries, click-throughs) to learn

about their preferences. Thus, it is often possible to bootstrap data-driven services by collecting explicit and implicit data from enthusiasts and early adopters of the system. The data collected from such a bootstrapping process not only enables more frequent and faster system deployment, but also better reflects the actual usage patterns of real users, critical to building effective data-driven applications.

A.2 GPS Navigation Service

As a specific example of a data-driven service based on community participation, consider a web-based GPS navigation service. Instead of collecting all navigational information and storing them on a static CD/DVD, this service allows users to access the most up-to-date information via the Internet. Thus, instead of selling the system for a fixed upfront fee, the company can generate a recurring revenue stream by charging a monthly fee for access to the navigational data.

Providing navigation information via the Internet addresses the problem of out-of-date road data. However, dispatching specially equipped vehicles to gather and maintain an accurate database of navigational information is time-consuming and costly. Instead, following the Web 2.0 model, such a service can potentially recruit users of the service to allow the system to automatically collect navigational data from the GPS unit in the background and send it to the service provider for analysis and system development. If the user is traveling on an unknown road, the system can explicitly request the user to provide additional information, such as the road name and speed limit.

This model reduces not only the cost of data collection, but also the update latency. Instead of ad hoc collection, the data collected via this approach are more likely to reflect the routes and traffic conditions that real users will experience. Furthermore, by enabling the collection of real-time information, the system can provide additional services such as real-time traffic and traffic-based trip planning. Finally, by analyzing the data from all participants over a period of time, machine learning techniques can be applied to predict traffic conditions based on historical data.

A.3 User Participation

The navigational services described above are only achievable with the participation of a community of users, essentially creating a network effect. Though, it is in the best interest of most users to participate in the data collection, some users are hesitant due to privacy concerns. Responding to explicit requests also requires additional work not all users are willing to put in. Thus, will we have enough user participation to bootstrap the system and offer the advanced services?

In Wikipedia, over half of the edits are made by less than 0.7% of registered users [147]. This extremely skewed distribution of edits suggests that even if an overwhelming majority of users declines to participate in the data collection effort, there will be super enthusiastic users who are willing to spend significant amount of effort to improve the quality of the system. By building a community where these enthusiasts can establish reputation, we may attract a sufficient number of early adopters not just as references, but to collaboratively develop the service and cross the chasm into the mass market.

Given the skew in user participation, a valid concern is that the content that users are willing to contribute may not always reflect the information others are seeking. For example,

most open source projects focus on applications that fulfill the needs of the developers, rarely niche markets with high research and development costs with few customers. However, by designing interfaces that lower the cost of participation, we can reduce the divergence in interest between the supply and the demand. In the case of GPS navigation systems, the automatically collection of navigational information in the background removes many of these differences. Live traffic information is generally needed only for frequently traveled roads, precisely the places where live navigational data are being collected from other users of the service.

A.4 Business Model

Traditionally, most GPS navigation systems are sold as customized hardware with embedded software for a fixed upfront fee ($400-$700). Consumers can obtain updated navigational data by purchasing new versions released every year or so ($100-$150). Some vendors also offer live traffic report on a subscription basis for select cities ($10/month).

With sufficient community participation, we can offer advanced navigational services that plan routes according to detailed real-time information on all roads, not just accident reports and coarse highway speeds. This information has the potential to significantly reduce travel times for motorists. Given the value placed on time, many people, especially businessmen working in cities, may be willing to subscribe to such a service for a substantial fee ($30/month?) if it can reduce their daily commute by 10 minutes. To promote data contribution, we can reduce the subscription fees and perhaps even subsidize the hardware cost for those users willing to share their GPS information. With this two-tier pricing model, privacy sensitive individuals can still enjoy the benefits of the services while the general population receives discounts in exchange of participation in the distributed data collection.

In addition to generating revenue from the consumers, the company can also sell the data to other GPS navigation vendors and other parties. This is the model employed by NAVTEQ, the leading provider of digital map data. With a growing revenues and profits exceeding $270 million in 2005, there is clearly a potential for significant profits to be made from collecting and selling data.

A.5 Conclusions

Most Web 2.0 ventures today focus on advertisement as the primary source of revenue. However, as we have demonstrated with a hypothetical GPS navigation service, there are significant revenue possibilities in the data contributed by the user communities formed around these services. As can be seen in the recent uproar over the release of AOL search queries, many users are still sensitive to privacy issues in the transfer of user data [85]. Perhaps through clearer licensing terms, two-tier service options, and better obfuscation of personally identifiable data through aggregation, businesses will be able resolve the privacy challenges and consider new business models that can leverage community participation for data collection.

Bibliography

[1] Michiel Bacchiani, Michael Riley, Brian Roark, and Richard Sproat. MAP adaptation of stochastic grammars. *Computer Speech & Language*, 20(1):41–68, 2006.

[2] Michiel Bacchiani and Brian Roark. Unsupervised language model adaptation. In *Proc. ICASSP*, Hong Kong, China, 2003.

[3] Lalit R. Bahl, Frederick Jelinek, and Robert L. Mercer. A maximum likelihood approach to continuous speech recognition. In *Readings in Speech Recognition*, pages 308–319. Morgan Kaufmann Publishers Inc., San Francisco, California, 1990.

[4] Michele Banko and Eric Brill. Scaling to very very large corpora for natural language disambiguation. In *Proc. ACL*, Toulouse, France, 2001.

[5] Jerome R. Bellegarda. Exploiting latent semantic information in statistical language modeling. *Proceedings of the IEEE*, 88(8):1279–1296, 2000.

[6] Jerome R. Bellegarda. Statistical language model adaptation: Review and perspectives. *Speech Communication*, 42(1):93–108, 2004.

[7] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.

[8] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[9] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proc. EMNLP*, Sydney, Australia, 2006.

[10] Thorsten Brants and Alex Franz. Web 1T 5-gram version 1. Linguistic Data Consortium, Philadelphia, 2006.

[11] Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. Large language models in machine translation. In *Proc. EMNLP-CoNNL*, Prague, Czech Republic, 2007.

[12] Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85, 1990.

[13] Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.

[14] Ivan Bulyko, Mari Ostendorf, Manhung Siu, Tim Ng, Andreas Stolcke, and Özgür Çetin. Web resources for language modeling in conversational speech recognition. *ACM Transactions on Speech and Language Processing*, 5(1):1–25, 2007.

[15] Ivan Bulyko, Mari Ostendorf, and Andreas Stolcke. Getting more mileage from web text sources for conversational speech language modeling using class-dependent mixtures. In *Proc. HLT*, Edmonton, Canada, 2003.

[16] Wray Buntine and Aleks Jakulin. Discrete principal component analysis. Technical report, Helsinki Institute for Information Technology, 2005.

[17] Mauro Cettolo, Fabio Brugnara, and Marcello Federico. Advances in the automatic transcription of lectures. In *Proc. ICASSP*, Montreal, Canada, 2004.

[18] Hung-An Chang and James Glass. Discriminative training of hierarchical acoustic models for large vocabulary continuous speech recognition. In *Proc. ICASSP*, Taipei, Taiwan, 2009.

[19] Ciprian Chelba and Alex Acero. Adaptation of maximum entropy capitalizer: Little data can help a lot. *Computer Speech and Language*, 20(4):382–399, 2006.

[20] Langzhou Chen, Jean-Luc Gauvain, Lori Lamel, and Gilles Adda. Unsupervised language model adaptation for broadcast news. In *Proc. ICASSP*, Hong Kong, China, 2003.

[21] Langzhou Chen and Taiyi Huang. An improved MAP method for language model adaptation. In *Proc. Eurospeech*, Budapest, Hungary, 1999.

[22] Stanley Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Computer Science Group, Harvard University, 1998.

[23] Stanley F. Chen and Ronald Rosenfeld. A Gaussian prior for smoothing maximum entropy models. Technical report, Carnegie Mellon University, 1999.

[24] Kenneth Church, Ted Hart, and Jianfeng Gao. Compressing trigram language models with Golomb coding. In *Proc. EMNLP-CoNLL*, Prague, Czech Republic, 2007.

[25] Philip R. Clarkson and A. J. Robinson. Language model adaptation using mixtures and an exponentially decaying cache. In *Proc. ICASSP*, Munich, Germany, 1997.

[26] Philip R. Clarkson and Roni Rosenfeld. Statistical language modeling using the CMU-Cambridge toolkit. In *Proc. Eurospeech*, Rhodes, Greece, 1997.

[27] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 2nd edition, 2001.

[28] Nils Dahlbäck, Arne Jönsson, and Lars Ahrenberg. Wizard of Oz studies: Why and how. In *Proc. Intelligent User Interfaces*, Orlando, Florida, USA, 1993.

[29] J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43(5):1470–1480, 1972.

[30] Hal Daumé III. Frustratingly easy domain adaptation. In *Proc. ACL*, Prague, Czech Republic, 2007.

[31] Hal Daumé III and Daniel Marcu. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26:101–126, 2006.

[32] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990.

[33] Arthur Dempster, Nan Laird, and Donald Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society, Series B (Methodological)*, 39(1):1–38, 1977.

[34] Paul G. Donnelly, F. J. Smith, E. Sicilia, and Ji Ming. Language modelling with hierarchical domains. In *Proc. Eurospeech*, Budapest, Hungary, 1999.

[35] Mark Dredze and Koby Crammer. Online methods for multi-domain learning and adaptation. In *Proc. EMNLP*, Honolulu, Hawaii, USA, 2008.

[36] Miroslav Dudík, David M. Blei, and Robert E. Schapire. Hierarchical maximum entropy density estimation. In *Proc. ICML*, Corvalis, Oregon, USA, 2007.

[37] Miroslav Dudík, Steven J. Phillips, and Robert E. Schapire. Maximum entropy density estimation with generalized regularization and an application to species distribution modeling. *Journal of Machine Learning Research*, 8:1217–1260, 2007.

[38] Marcello Federico. Bayesian estimation methods for n-gram language model adaptation. In *Proc. ICSLP*, Philadelphia, Pennsylvania, USA, 1996.

[39] Marcello Federico. Efficient language model adaptation through MDI estimation. In *Proc. Eurospeech*, Budapest, Hungary, 1999.

[40] Marcello Federico. Language model adaptation through topic decomposition and MDI estimation. In *Proc. of ICASSP*, Orlando, Florida, USA, 2002.

[41] Marcello Federico, Nicola Bertoldi, and Mauro Cettolo. IRSTLM: An open source toolkit for handling large scale language models. In *Proc. Interspeech*, Brisbane, Australia, 2008.

[42] Marcello Federico and Mauro Cettolo. Efficient handling of n-gram language models for statistical machine translation. In *Workshop on Statistical Machine Translation*, Prague, Czech Republic, 2007.

[43] Alexander Franz and Brian Milch. Searching the web by voice. In *Proc. COLING*, Taipei, Taiwan, 2002.

[44] Atsushi Fujii, Katunobu Itou, Tomoyosi Akiba, and Tetsuya Ishikawa. Unsupervised topic adaptation for lecture speech retrieval. In *Proc. Interspeech*, Jeju Island, Korea, 2004.

[45] Sadaoki Furui. Recent advances in spontaneous speech recognition and understanding. In *Proc. IEEE Workshop on Spontaneous Speech Processing and Recognition*, Tokyo, Japan, 2003.

[46] Jianfeng Gao and Kai-Fu Lee. Distribution-based pruning of backoff language models. In *Proc. ACL*, Hong Kong, China, 2000.

[47] Jianfeng Gao, Hai-Feng Wang, Mingjing Li, and Kai-Fu Lee. A unified approach to statistical language modeling for Chinese. In *Proc. ICASSP*, Istanbul, Turkey, 2000.

[48] Jean-Luc Gauvain, Lori Lamel, and Gilles Adda. The LIMSI broadcast news transcription system. *Speech Communication*, 37(1-2):89–108, 2002.

[49] Daniel Gildea and Thomas Hofmann. Topic-based language model using EM. In *Proc. Eurospeech*, Budapest, Hungary, 1999.

[50] Jean-Luc Gauvain Gilles Adda, Michèle Jardino. Language modeling for broadcast news transcription. In *Proc. Eurospeech*, Budapest, Hungary, 1999.

[51] Laurence Gillick and Stephen Cox. Some statistical issues in the comparison of speech recognition algorithms. In *Proc. ICASSP*, Glasgow, UK, 1989.

[52] Diego Giuliani and Marcello Federico. Unsupervised language and acoustic model adaptation for cross domain portability. In *Proc. ISCA Workshop on Adaptation Methods for Speech Recognition*, Sophia-Antipolis, France, 2001.

[53] James Glass. Challenges for spoken dialogue systems. In *Proc. ASRU*, 1999.

[54] James Glass. A probabilistic framework for segment-based speech recognition. *Computer Speech & Language*, 17(2-3):137–152, 2003.

[55] James Glass, Timothy Hazen, Scott Cyphers, Igor Malioutov, David Huynh, and Regina Barzilay. Recent progress in the MIT spoken lecture processing project. In *Proc. Interspeech*, Antwerp, Belgium, 2007.

[56] James Glass, Timothy J. Hazen, Lee Hetherington, and Chao Wang. Analysis and processing of lecture audio data: Preliminary investigations. In *Proc. HLT-NAACL Workshop on Interdisciplinary Approaches to Speech Indexing and Retrieval*, Boston, Massachusetts, USA, 2004.

[57] James Glass, Joe Polifroni, Stephanie Seneff, and Victor Zue. Data collection and performance evaluation of spoken dialogue systems: The MIT experience. In *Proc. Interspeech*, Beijing, China, 2000.

[58] John J. Godfrey and Edward Holliman. Switchboard-1 transcripts. Linguistic Data Consortium, Philadelphia, 1993.

[59] Joshua Goodman. Exponential priors for maximum entropy models. In *Proc. HLT-HAACL*, Boston, Massachusetts, USA, 2004.

[60] Joshua T. Goodman. A bit of progress in language modeling. *Computer Speech & Language*, 15(4):403–434, 2001.

[61] Google. Google sets. `http://labs.google.com/sets`, 2007.

[62] Google. Google audio indexing. `http://labs.google.com/gaudi`, 2008.

[63] David Graff. 1996 English broadcast news speech (HUB4). Linguistic Data Consortium, Philadelphia, 1997.

[64] David Graff. English Gigaword. Linguistic Data Consortium, Philadelphia, 2003.

[65] Roberto Gretter and Giuseppe Riccardi. On-line learning of language models with word error probability distributions. In *Proc. ICASSP*, Salt Lake City, Utah, USA, 2001.

[66] Thomas Griffiths and Mark Steyvers. Finding scientific topics. In *Proceedings of the National Academy of Sciences, 101 (suppl. 1)*, 2004.

[67] Thomas L. Griffiths, Mark Steyvers, David M. Blei, and Joshua B. Tenenbaum. Integrating topics and syntax. In *Proc. Advanced in Neural Information Processing Systems*, Vancouver, British Columbia, Canada, 2004.

[68] Timothy J. Hazen. Automatic alignment and error correction of human generated transcripts for long speech recordings. In *Proc. Interspeech*, Pittsburgh, Pennsylvania, USA, 2006.

[69] Thomas Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning Journal*, 42(1):177–196, 2001.

[70] Bo-June (Paul) Hsu. Generalized linear interpolation of language models. In *Proc. ASRU*, Kyoto, Japan, 2007.

[71] Bo-June (Paul) Hsu and James Glass. Spoken correction for Chinese text entry. In *Proc. Chinese Spoken Language Processing*, Kent Ridge, Singapore, 2006.

[72] Bo-June (Paul) Hsu and James Glass. Style & topic language model adaptation using HMM-LDA. In *Proc. EMNLP*, Sydney, Australia, 2006.

[73] Bo-June (Paul) Hsu and James Glass. Iterative language model estimation: Efficient data structure & algorithms. In *Proc. Interspeech*, Brisbane, Australia, 2008.

[74] Bo-June (Paul) Hsu and James Glass. N-gram weighting: Reducing training data mismatch in cross-domain language model estimation. In *Proc. EMNLP*, Honolulu, Hawaii, USA, 2008.

[75] Bo-June (Paul) Hsu and James Glass. Language model parameter estimation using user transcriptions. In *Proc. ICASSP*, Taipei, Taiwan, 2009.

[76] Bo-June (Paul) Hsu, Milind Mahajan, and Alex Acero. Multimodal text entry on mobile devices. In *Demo at ASRU*, San Juan, Puerto Rico, 2005.

[77] Xuedong Huang, Alex Acero, and Hsiao-Wuen Hon. *Spoken Language Processing – A Guide to Theory, Algorithms, and System Development*. Prentice Hall, New York, New York, USA, 2001.

[78] David Huggins-Daines and Alexander I. Rudnicky. Interactive ASR error correction for touchscreen devices. In *Proc. ACL-HLT*, Columbus, Ohio, USA, 2008.

[79] Rukmini M. Iyer and Mari Ostendorf. Modeling long distance dependence in language: Topic mixtures versus dynamic cache models. *IEEE Transactions on Speech and Audio Processing*, 7(1):30–39, 1999.

[80] Michèle Jardino. Multilingual stochastic n-gram class language models. In *Proc. ICASSP*, Atlanta, Georgia, USA, 1996.

[81] Frederick Jelinek and Robert L. Mercer. Interpolated estimation of Markov source parameters from sparse data. In *Proc. Workshop on Pattern Recognition in Practice*, Amsterdam, Netherlands, 1980.

[82] Nikola Jevtić and Alon Orlitsky. A universal compression perspective of smoothing, 2004. Submitted to *Computer Speech & Language*.

[83] Kazuomi Kato, Hiroaki Nanjo, and Tatsuya Kawahara. Automatic transcription of lecture speech using topic-independent language modeling. In *Proc. Interspeech*, Beijing, China, 2000.

[84] Slava M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(3):400–401, 1987.

[85] Dawn Kawamoto and Elinor Mills. AOL apologizes for release of user search data. `http://news.cnet.com/2100-1030_3-6102793.html`, August 2006.

[86] Jun'Ichi Kazama and Jun'Ichi Tsujii. Maximum entropy models with inequality constraints: A case study on text categorization. *Machine Learning*, 60(1-3):159–194, 2005.

[87] Frank Keller, Maria Lapata, and Olga Ourioupina. Using the web to overcome data sparseness. In *Proc. EMNLP*, Philadelphia, Pennsylvania, USA, 2002.

[88] Sanjeev Khudanpur and Jun Wu. Maximum entropy techniques for exploiting syntactic, semantic and collocational dependencies in language modeling. *Computer Speech & Language*, 14(4):355–372, 2000.

[89] Dietrich Klakow. Log-linear interpolation of language models. In *Proc. ICSLP*, Sydney, Australia, 1998.

[90] Reinhard Kneser and Hermann Ney. Improved backing-off for m-gram language modeling. In *Proc. ICASSP*, Detroit, Michigan, USA, 1995.

[91] Reinhard Kneser, Jochen Peters, and Dietrich Klakow. Language model adaptation using dynamic marginals. In *Proc. Eurospeech*, Rhodes, Greece, 1997.

[92] Rony Kubat, Philip DeCamp, Brandon Roy, and Deb Roy. TotalRecall: Visualization and semi-automatic annotation of very large audio-visual corpora. In *Proc. International Conference on Multimodal Interfaces*, Antwerp, Belgium, 2007.

[93] Roland Kuhn and Renato De Mori. A cache-based natural language model for speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):570–583, 1990.

[94] Julian Kupiec. Robust part-of-speech tagging using a hidden Markov model. *Computer Speech & Language*, 6(3):225–242, 1992.

[95] Mirella Lapata and Frank Keller. Web-based models for natural language processing. *ACM Transactions on Speech and Language Processing*, 2(1):1–30, 2005.

[96] Raymond Lau, Ronald Rosenfeld, and Salim Roukos. Trigger-based language models: A maximum entropy approach. In *Proc. ICASSP*, Minneapolis, Minnesota, USA, 1993.

[97] Erwin Leeuwis, Marcello Federico, and Mauro Cettolo. Language modeling and transcription of the TED corpus lectures. In *Proc. ICASSP*, Hong Kong, China, 2003.

[98] Michael Lehn. Everything you always wanted to know about FLENS, but were afraid to ask, February 2008. `http://flens.sf.net/flensdoc.pdf`.

[99] Xiao Li, Yun-Cheng Ju, Geoff Zweig, and Alex Acero. Language modeling for voice search: A machine translation approach. In *Proc. ICASSP*, Las Vegas, Nevada, USA, 2008.

[100] Xinwei Li, Hui Jiang, and Chaojun Liu. Large margin HMMs for speech recognition. In *Proc. ICASSP*, Philadelphia, Pennsylvania, USA, 2005.

[101] Yang Liu and Feifan Liu. Unsupervised language model adaptation via topic modeling based on named entity hypotheses. In *Proc. ICASSP*, Las Vegas, Nevada, USA, 2008.

[102] Kikuo Maekawa, Hanae Koiso, Sadaoki Furui, and Hitoshi Isahara. Spontaneous speech corpus of Japanese. In *Proc. Language Resources and Evaluation*, Athens, Greece, 2000.

[103] Igor Malioutov and Regina Barzilay. Minimum cut model for spoken lecture segmentation. In *Proc. ACL*, Sydney, Australia, 2006.

[104] Robert Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *Proc. COLING*, Taipei, Taiwan, 2002.

[105] G. Maltese, P. Bravetti, Hubert Crépy, B. J. Grainger, M. Herzog, and Francisco Palou. Combining word- and class-based language models: A comparative study in several languages using automatic and manual word-clustering techniques. In *Proc. Interspeech*, Geneva, Switzerland, 2001.

[106] Urs-Viktor Marti and Horst Bunke. Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(1):65–90, 2001.

[107] Hirokazu Masataki, Yoshinori Sagisaka, Kazuya Hisaki, and Tatsuya Kawahara. Task adaptation using MAP estimation in n-gram language modeling. In *Proc. ICASSP*, Munich, Germany, 1997.

[108] Erik McDermott, Timothy J. Hazen, Jonathan Le Roux, Atsushi Nakamura, and Shigeru Katagiri. Discriminative training for large-vocabulary speech recognition using minimum classification error. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(1):203–223, 2007.

[109] Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88, 2002.

[110] Cosmin Munteanu, Gerald Penn, and Ron Baecker. Web-based language modelling for automatic lecture transcription. In *Proc. Interspeech*, Antwerp, Belgium, 2007.

[111] Hiroaki Nanjo and Tatsuya Kawahara. Unsupervised language model adaptation for lecture speech recognition. In *Proc. ISCA & IEEE Workshop on Spontaneous Speech Processing and Recognition*, Tokyo, Japan, 2003.

[112] Hiroaki Nanjo and Tatsuya Kawahara. Language model and speaking rate adaptation for spontaneous presentation speech recognition. *IEEE Transactions on Speech and Audio Processing*, 12(4):391–400, 2004.

[113] Hermann Ney, Ute Essen, and Reinhard Kneser. On structuring probabilistic dependences in stochastic language modelling. *Computer Speech & Language*, 8(1):1–38, 1994.

[114] Patrick Nguyen, Jianfeng Gao, and Milind Mahajan. MSRLM: A scalable language modeling toolkit. Technical Report MSR-TR-2007-144, Microsoft, Inc., 2007.

[115] Thomas Niesler and Daniel Willett. Unsupervised language model adaptation for lecture speech transcription. In *Proc. Interspeech*, Denver, Colorado, USA, 2002.

[116] Miroslav Novak and Richard Mammone. Use of non-negative matrix factorization for language model adaptation in a lecture transcription task. In *Proc. ICASSP*, Salt Lake City, Utah, USA, 2001.

[117] Rasmus Pagh and Flemming Friche Rodler. Cuckoo hashing. *Journal of Algorithms*, 51(2):122–144, 2004.

[118] David S. Pallett, William M. Fisher, and Jonathan G. Fiscus. Tools for the analysis of benchmark speech recognition tests. In *Proc. ICASSP*, Alburquerque, New Mexico, USA, 1990.

[119] Alex Park, Timothy J. Hazen, and James Glass. Automatic processing of audio lectures for information retrieval: Vocabulary selection and language modeling. In *Proc. ICASSP*, Philadelphia, Pennsylvania, USA, 2005.

[120] Fernando Pereira. Multinomial logistic regression/maximum entropy. `http://www.cis.upenn.edu/~pereira/classes/CIS620/lectures/maxent.pdf`, January 2005.

[121] Stephen Della Pietra, Vincent Della Pietra, Robert L. Mercer, and Salim Roukos. Adaptive language modeling using minimum discriminant estimation. In *Proc. ICASSP*, San Francisco, California, USA, 1992.

[122] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *Proc. SIGIR*, Melbourne, Australia, 1998.

[123] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes*. Cambridge University Press, 3rd edition, 2007.

[124] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[125] P. Srinivasa Rao, Satya Dharanipragada, and Salim Roukos. MDI adaptation of language models across corpora. In *Proc. Eurospeech*, Rhodes, Greece, 1997.

[126] Wolfgang Reichl. Language model adaptation using minimum discrimination information. In *Proc. Eurospeech*, Budapest, Hungary, 1999.

[127] Giuseppe Riccardi and Dilek Hakkani-Tür. Active learning: Theory and applications to automatic speech recognition. *IEEE Transactions on Speech and Audio Processing*, 13(4):504–511, 2005.

[128] Michal Rosen-Zvi, Thomas Griffiths, Mark Steyvers, and Padhraic Smyth. The author-topic model for authors and documents. In *Proc. Uncertainty in Artificial Intelligence*, Banff, Canada, 2004.

[129] Roni Rosenfeld. A maximum entropy approach to adaptive statistical language modeling. *Computer Speech & Language*, 10(3):187–228, 1996.

[130] Roni Rosenfeld. The "degenerate EM" algorithm for finding optimal linear interpolation coefficients λ_i, February 2007. `http://www.cs.cmu.edu/~roni/11761-s07/Presentations/degenerateEM.pdf`.

[131] Abhinav Sethy, Panayiotis Georgiou, and Shrikanth Narayanan. Building topic specific language models from webdata using competitive models. In *Proc. Interspeech*, Lisboa, Portugal, 2005.

[132] Abhinav Sethy, Panayiotis Georgiou, Bhuvana Ramabhadran, and Shrikanth Narayanan. An iterative relative entropy minimization based data selection approach for n-gram model adaptation. *IEEE Transactions on Speech, Audio and Language Processing*, 17(1):13–23, 2009.

[133] Abhinav Sethy, Panayiotis G. Georgiou, and Shrikanth Narayanan. Text data acquisition for domain-specific language models. In *Proc. EMNLP*, Sydney, Australia, 2006.

[134] Kristie Seymore and Ronald Rosenfeld. Using story topics for language model adaptation. In *Proc. Eurospeech*, Rhodes, Greece, 1997.

[135] Bernd Souvignier, Andreas Kellner, Bernhard Rueber, Hauke Schramm, and Frank Seide. The thoughtful elephant: Strategies for spoken dialog systems. *IEEE Transactions on Speech and Audio Processing*, 8(1):51–62, 2000.

[136] Richard Sproat, Alan W. Black, Stanley Chen, Shankar Kumar, Mari Ostendorf, and Christopher Richards. Normalization of non-standard words. *Computer Speech & Language*, 15(3):287–333, 2001.

[137] Michael Steinbach, George Karypis, and Vipin Kumar. A comparison of document clustering techniques. Technical Report #00-034, University of Minnesota, 2000.

[138] Andreas Stolcke. SRILM – An extensible language modeling toolkit. In *Proc. Interspeech*, Denver, Colorado, USA, 2002.

[139] Andreas Stolcke. SRILM man pages: ngram-format, 2004. `http://www.speech.sri.com/projects/srilm/manpages/ngram-format.html`.

[140] David Talbot and Thorsten Brants. Randomized language models via perfect hash functions. In *Proc. ACL-HLT*, Columbus, Ohio, USA, 2008.

[141] Yik-Cheung Tam and Tanja Schultz. Unsupervised language model adaptation using latent semantic marginals. In *Proc. Interspeech*, Pittsburgh, Pennsylvania, USA, 2006.

[142] Yee Whye Teh. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proc. ACL*, Sydney, Australia, 2006.

[143] Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

[144] Gokhan Tur and Andreas Stolcke. Unsupervised language model adaptation for meeting recognition. In *Proc. ICASSP*, Honolulu, Hawaii, USA, 2007.

[145] David Vadas and James Curran. Adding noun phrase structure to the Penn treebank. In *Proc. ACL*, Prague, Czech Republic, 2007.

[146] Todd Veldhuizen. Expression templates. *C++ Report*, 7:26–31, 1995.

[147] Jimmy Wales. Wikipedia: Social innovation on the web. In *Proc. Web Design for Interactive Learning*, Ithaca, New York, USA, 2005.

[148] Richard C. Wang and William W. Cohen. Language-independent set expansion of named entities using the web. In *Proc. Data Mining*, Omaha, Nebraska, USA, 2007.

[149] Ye-Yi Wang, Li Deng, and Alex Acero. Spoken language understanding – An introduction to the statistical framework. *IEEE Signal Processing Magazine*, 22(5):16–31, 2005.

[150] Edward Whittaker and Bhiksha Raj. Quantization-based language model compression. In *Proc. Interspeech*, Aalborg, Denmark, 2001.

[151] Daniel Willett, Thomas Niesler, Erik McDermott, Yasuhiro Minami, and Shigeru Katagiri. Pervasive unsupervised adaptation for lecture speech transcription. In *Proc. ICASSP*, Hong Kong, China, 2003.

[152] Ian H. Witten and Timothy C. Bell. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4):1085–1094, 1991.

[153] George Y. Wong and William M. Mason. The hierarchical logistic regression model for multilevel analysis. *Journal of the American Statistical Association*, 80(391):513–524, 1985.

[154] Hui Ye and Steve Young. A clustering approach to semantic decoding. In *Proc. Interspeech*, Pittsburgh, Pennsylvania, USA, 2006.

[155] Tadasuke Yokoyama, Takahiro Shinozaki, Koji Iwano, and Sadaoki Furui. Unsupervised class-based language model adaptation for spontaneous speech recognition. In *Proc. ICASSP*, Hong Kong, China, 2003.

[156] Dong Yu, Li Deng, Xiaodong He, and Alex Acero. Large-margin minimum classification error training for large-scale speech recognition tasks. In *Proc. ICASSP*, Honolulu, Hawaii, USA, 2007.

[157] Zheng-Yu Zhou, Jian-Feng Gao, and Eric Chang. Improving language modeling by combining heteogeneous corpora. In *Proc. Chinese Spoken Language Processing*, Taipei, Taiwan, 2002.

[158] Zheng-Yu Zhou, Peng Yu, Ciprian Chelba, and Frank Seide. Towards spoken-document retrieval for the Internet: Lattice indexing for large-scale web-search architectures. In *Proc. HLT-NAACL*, New York, New York, USA, 2006.

[159] Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, 1997.

[160] Xiaojin Zhu and Ronald Rosenfeld. Improving trigram language modeling with the World Wide Web. In *Proc. ICASSP*, Salt Lake City, Utah, USA, 2001.

[161] Imed Zitouni. Backoff hierarchical class n-gram language models: Effectiveness to model unseen events in speech recognition. *Computer Speech & Language*, 21(1):88–104, 2007.

[162] Victor Zue, Stephanie Seneff, James Glass, Joe Polifroni, Timothy Hazen, and Lee Hetherington. JUPlTER: A telephone-based conversational interface for weather information. *IEEE Transactions on Speech and Audio Processing*, 8(1):85–96, 2000.